# GEO v.0.0.0

17 de marzo de 2009

**GE**odesia para **O**ctave
**GE**odesy for **O**ctave
**GE**odäsie für **O**ctave
**GÉ**odésie pour **O**ctave
**GE**odesia per **O**ctave
**GE**odesie voor **O**ctave

## Manual de referencia

**José Luis García Pallero**

# Proyecto GEO

*Copyright* © 2005, 2006, 2007, 2008, 2009
José Luis García Pallero, jgpallero@gmail.com

**Aviso legal. Sobre este documento**

Se garantiza permiso para copiar, distribuir y modificar este documento según los términos de la *GNU Free Documentation License* (FDL), v.1.3 o cualquiera posterior publicada por la *Free Software Foundation* (FSF), sin secciones invariantes ni textos de cubierta delantera o trasera. Puede obtener una copia de la FDL o ponerse en contacto con la FSF a través de las siguientes direcciones web: http://www.fsf.org o http://www.gnu.org. Puede obtener una copia transparente de este documento (en formato LaTeX) a través de jgpallero@gmail.com o bien puede generarla con las funciones geo_GeneraAyudaLaTeX.m, geo_GeneraAyudaUsuarioLaTeX.m o geo_GeneraAyudaFuncionesLaTeX.m de GEO.

**Aviso legal. Sobre las funciones de GEO**

El proyecto GEO es *software* libre. Usted puede distribuirlo y/o modificarlo bajo los términos de la *GNU General Public License* (GPL), v.3 o cualquiera posterior publicada por la *Free Software Foundation* (FSF). Puede obtener una copia de la GPL o ponerse en contacto con la FSF a través de las siguientes direcciones web: http://www.fsf.org o http://www.gnu.org.

# Índice general

**Índice general** 3

**Índice de figuras** 7

**Índice de cuadros** 9

**Introducción** 11

**Por hacer** 13

**Evolución de GEO** 15

**Errores (*bugs*) de GEO** 17

## I Manual del usuario 19

**1. La variable global `geoVarGlobal`** 21
- 1.1. Unidades angulares . . . 21
- 1.2. Dominios geodésicos . . . 21

**2. Tratamiento de modelos de geoide** 23

**3. Herramientas de dibujo** 25

**4. Generación automática de la documentación** 27
- 4.1. Tipos de documentación . . . 27
- 4.2. Generación de los ficheros fuente . . . 27
- 4.3. Compilación de los ficheros fuente . . . 28
  - 4.3.1. Compilación automática . . . 28
  - 4.3.2. Compilación manual . . . 28

## II Manual del desarrollador 29

**5. Cómo escribir código fuente para GEO** 31
- 5.1. Cómo escribir funciones en lenguaje *GNU Octave* . . . 31
- 5.2. Cómo escribir funciones en lenguaje *C/C++* . . . 31
- 5.3. Cómo escribir código *C/C++* para la biblioteca `libgeo.a` . . . 31

**6. Documentación de la biblioteca GEO** 33

**7. Propagación de errores en GEO** 35

**8. Optimización de la velocidad de ejecución del código** 37

## III Ayuda de las funciones 39

**9. Sección** `01-global` 41

**10. Sección** `02-general` 43
10.1. Carpeta `01-controlTipoDato` . . . 43
10.2. Carpeta `02-controlDominio` . . . 48
10.3. Carpeta `03-angulos` . . . 52
10.4. Carpeta `04-geoLista` . . . 57
10.5. Carpeta `05-matrices` . . . 61
10.5.1. Carpeta `01-matricesGeneral` . . . 61
10.5.2. Carpeta `02-matricesVector` . . . 66
10.6. Carpeta `06-estadistica` . . . 71
10.7. Carpeta `07-elipsesError` . . . 76
10.8. Carpeta `08-datosBinarios` . . . 77
10.8.1. Carpeta `01-tipoDatoBinario` . . . 77
10.8.2. Carpeta `02-conversorBinario` . . . 80
10.9. Carpeta `09-interpMalla` . . . 82
10.10 Carpeta `10-interpNubePuntos` . . . 87
10.11 Carpeta `11-delaunay` . . . 94
10.12 Carpeta `12-sistEcuaciones` . . . 97
10.12.1 Carpeta `01-mmcc` . . . 97
10.12.2 Carpeta `02-metodosRobustos` . . . 102
10.13 Carpeta `13-dibujo` . . . 102
10.13.1 Carpeta `01-estructuraDibujo` . . . 102
10.13.2 Carpeta `02-dibujoOctave` . . . 119
10.13.3 Carpeta `03-dibujoDxf` . . . 131
10.14 Carpeta `14-funcionesCpp` . . . 144
10.14.1 Carpeta `01-fuentes` . . . 144
10.14.1.1 Carpeta `01-fuentesGeneral` . . . 144
10.14.1.1.1. Carpeta `01-h` . . . 144
10.14.1.1.2. Carpeta `02-cc` . . . 147
10.14.1.2 Carpeta `02-fuentesOct` . . . 149
10.14.2 Carpeta `02-oct` . . . 162
10.14.3 Carpeta `03-compilacion` . . . 162
10.15 Carpeta `15-generadorAyuda` . . . 162
10.16 Carpeta `16-informacion` . . . 166
10.17 Carpeta `17-otras` . . . 168
10.18 Carpeta `18-nuevas` . . . 194
10.19 Carpeta `19-viejas` . . . 194

**11. Sección** `03-elipsoide` 197

**12. Sección** `04-geoide` 199
12.1. Carpeta `01-estructuraGeoide` . . . 199
12.2. Carpeta `02-ondulacionGeoide` . . . 201

**13.Sección** 05-mde **203**
13.1. Carpeta 01-estructuraMde . . . 203
13.2. Carpeta 02-altitudMde . . . 205

**14.Sección** 06-calculosGeodesicos **209**
14.1. Carpeta 01-dominiosGeodesicos . . . 209
14.2. Carpeta 02-transformacionSistCoord . . . 210
14.3. Carpeta 03-latitudes . . . 215
14.4. Carpeta 04-calculosPlano . . . 223
14.5. Carpeta 05-calculosEspacio . . . 242
14.6. Carpeta 06-calculosEsfera . . . 243
14.7. Carpeta 07-calculosElipsoide . . . 247

**15.Sección** 07-proyCarto **259**
15.1. Carpeta 01-estructuraProyCarto . . . 259
15.2. Carpeta 02-transformacionProyCarto . . . 259
15.3. Carpeta 03-proyecciones . . . 259
15.3.1. Carpeta estereografica . . . 259
15.3.2. Carpeta tranMerc . . . 262

**16.Sección** 08-redesGeodesicas **267**
16.1. Carpeta 01-disenyoRedes . . . 267
16.1.1. Carpeta 01-generacionObsRedes . . . 267
16.2. Carpeta 02-analisisRedes . . . 268
16.3. Carpeta 03-calculoRedes . . . 270
16.4. Carpeta 04-compensacionRedes . . . 274

**17.Sección** 09-datum **279**

**18.Sección** 10-gravimetria **281**
18.1. Carpeta 01-armonicosEsfericos . . . 281
18.2. Carpeta 02-gravedad . . . 283
18.3. Carpeta 03-probInvGrav . . . 295
18.4. Carpeta 04-obsGravedad . . . 296

**19.Sección** 11-astronomia **299**
19.1. Carpeta 01-tiempo . . . 299
19.2. Carpeta 02-gps . . . 300
19.3. Carpeta 03-efemerides . . . 300

**20.Sección** 12-instrumento **303**
20.1. Carpeta 01-estructuraInstrumento . . . 303
20.2. Carpeta 02-calculosInstrumento . . . 303

**21.Sección** 13-datos **305**

**22.Sección** 14-pruebas **317**

**23.Sección** 15-documentacion **327**
23.1. Carpeta 01-ayudaPdf . . . 327
23.2. Carpeta 02-ayudaHtml . . . 327
23.3. Carpeta 03-definicionSintaxis . . . 327
23.4. Carpeta 04-fuentesAyuda . . . 327
23.4.1. Carpeta 01-ficheroGlobal . . . 327

23.4.2. Carpeta 02-preambulo . . . 327
23.4.3. Carpeta 03-frontmatter . . . 327
23.4.4. Carpeta 04-mainmatter . . . 328
23.4.4.1. Carpeta 01-ficheroGlobalMainmatter . . . 328
23.4.4.2. Carpeta 02-fuentesUsuario . . . 328
23.4.4.3. Carpeta 03-fuentesDesarrollador . . . 328
23.4.4.4. Carpeta 04-fuentesIntroduccionCarpetas . . . 328
23.4.4.5. Carpeta 05-fuentesApendices . . . 328
23.4.4.6. Carpeta 06-fuentesLicencias . . . 328
23.4.5. Carpeta 05-backmatter . . . 328
23.4.6. Carpeta 06-auxiliares . . . 328
23.4.6.1. Carpeta 01-bibliografia . . . 328
23.4.6.2. Carpeta 02-figuras . . . 328
23.4.6.3. Carpeta 03-estilos . . . 328
23.4.6.4. Carpeta 04-makefiles . . . 329
23.5. Carpeta 05-licencias . . . 329

**IV Apéndices** **331**

**A. Instalación de *GNU Octave*** **333**

**B. Editores de texto** **335**
B.1. El editor *Kate* . . . 335
B.1.1. Instalación de archivos de resaltado de sintaxis . . . 335
B.2. El editor *vi/vim* . . . 335
B.2.1. Instalación de archivos de resaltado de sintaxis . . . 335
B.3. El editor *CodePad* . . . 335
B.4. El editor *nano* . . . 335
B.5. El editor *Gedit* . . . 335
B.6. El editor *jEdit* . . . 336
B.7. El editor *emacs* . . . 336
B.8. El editor *Smultron* . . . 336
B.9. El editor *Geany* . . . 336
B.10. El editor *SciTE* . . . 336

**C. GNU General Public License** **337**

**D. GNU Free Documentation License** **347**

**Bibliografía** **355**

**Índice terminológico** **357**

**Índice de funciones** **359**

**Índice de funciones compiladas** **363**

**Índice de ficheros C/C++, Fortran y Bash** **365**

**Índice de ficheros de datos** **367**

**Índice de programas de prueba** **369**

# Índice de figuras

# Índice de cuadros

# Introducción

## El proyecto GEO

El proyecto GEO nace con dos objetivos fundamentales a alcanzar:

- Crear un conjunto de herramientas informáticas capaces de resolver los problemas que plantean la geodesia, la topografía y la cartografía matemática.
- Que estas herramientas estén a disposición de cualquier individuo y que éste, a su vez, tenga total libertad para utilizar, distribuir y ampliar el conjunto; esto es, que GEO entre a formar parte de lo que se ha dado en llamar *software* libre.

## Acerca de las normas ortográficas del español en GEO

En la escritura del código (ayuda y comentarios) de las funciones y ficheros de configuración de GEO se han omitido algunas reglas de puntuación (tildes y diéresis) del idioma español, así como la sustitución de la letra «ñ» por la «n».

Estas medidas han sido tomadas como previsión ante posibles errores de presentación en pantalla debidos a la utilización de distintos juegos de caracteres por parte de los múltiples sistemas informáticos existentes. Si bien es cierto que la mayor parte de los editores de texto dan la posibilidad de elegir el juego de caracteres utilizado, con lo que se solucionaría el problema de la correcta visualización durante la programación, nada garantiza que durante el trabajo interactivo en *GNU Octave* o *MatLab* la presentación de la ayuda de las funciones en pantalla mediante el comando `help` sea correcta.

Sirvan los párrafos anteriores de justificación ante el hecho de la omisión en GEO de las normas ortográficas del español citadas.

# Por hacer

1. Aclarar en la ayuda de las funciones de gravedad el signo del resultado.
2. Actualizar la generación automática de la ayuda de la biblioteca.
3. En las funciones de interpolación de datos ponderando con el inverso de la distancia hay que tener en cuenta el caso particular de que una sea 0.
4. En las interpolaciones de datos en mallas cuadradas hay que tener en cuenta la posibilidad de nodos con valor desconocido y la posibilidad de repetición o no del primer dato en longitud para mallas con cobertura global.
5. Comprobar que el tamaño (filas) de las matrices de varianza-covarianza es congruente con el tamaño de las observaciones a las que van asociadas.
6. Optimizar el uso de la funcion `geo_AjustaFilasVectorMatriz` cuando es llamada muchas veces en una funcion.
7. En la función `geo_IntTresDistanciasPlano.m` queda por introducir la información de los errores de los puntos fijos en el ajuste. Tal y como está funciona perfectamente, pero sólo se pondera el ajuste con la información de los errores en las distancias medidas.
8. Introducir la función de mínimos cuadrados en la función `geo_IntTresDistanciasPlano.m`.
9. Crear programa de prueba para la intersección de distancias.
10. Las funciones `geo_GenRedGps.m` y `geo_GenRedPlanoCla.m` no están programadas completamente en el sentido de que falta por programar la parte de instrumentación topográfica.
11. Optimizar las funciones `geo_ProbDirGeoElip` y `geo_ProbInvGeoElip` en sus versiones C++ y *GNU Octave*.
12. Modificar el *script* `compilaOct.sh` para que verifique la existencia y los permisos del directorio donde se guardarán las funciones compiladas.
13. Eliminar la duplicidad de código de las parejas de funciones:
    `geo_MatrizSimVector.m`
    `geo_MatrizSimVectorInterna.m`
    y
    `geo_VectorMatrizSim.m`
    `geo_VectorMatrizSimInterna.m`.
14. Revisar todas aquellas funciones en las que los argumentos opcionales puedan tomar varios valores (especialmente cuando sean cadenas de texto). Sería conveniente que se efectuase el chequeo del valor del argumento en el cuerpo de ejecución de la función.
15. Crear programas de prueba para las funciones de cálculo del radio de curvatura de la elipse meridiana, del radio de curvatura del primer vertical y del radio de curvatura de un paralelo.

# Evolución de GEO

A continuación se listan los cambios que ha ido sufriendo GEO a lo largo del tiempo. Si modifica o añade alguna función o sección, indique el cambio en este apartado.

1. 25-12-06, versión 0.0.0: Estoy reorganizando la generación de ayuda.

# Errores (*bugs*) de GEO

A continuación se listan los errores conocidos de GEO. Si encuentra alguno nuevo, añádalo a la lista; si corrige alguno, elimínelo.

1. Repasar la programación de la función `geo_GravedadPrismaMacMillan.m`. Parece que hay un error en cuanto al orden de las coordenadas $z$ del prisma atrayente. Comprobar con los datos:

```
pto = [150 1600 400];
pri = [100 200 100 3100 -600 -100];
ro = 500;
```

2. Las funciones `geo_ProbDirGeoEsf.m` y `geo_ProbDirGeoEsf.m` se vuelven inestables cuando trabajan con puntos cuya longitud es exactamente 180 grados y con puntos con latitudes cercanas a 90 grados.

3. Revisar las funciones `geo_ProbDirGeoElip` y `geo_ProbInvGeoElip` para que funcione bien el programa de prueba. Se producen errores cuando el punto de partida tiene una longitud exacta de 180 grados.

4. Hay una gran diferencia en el cálculo del error propagado en la función `geo_SuperficiePlano.m` según se tome matriz diagonal o completa en el error de los puntos dato.

5. Con respecto a las funciones de escritura de archivos DXF:

   *a*) No funcionan las órdenes `$EXTMIN` ni `$EXTMAX` que, en teoría, deberían definir los límites de visualización del dibujo. No sé si es porque la versión 12 de DXF no soporta esta característica o porque yo no he sabido asignarlas en el lugar correcto.

   *b*) No funciona la definición de tipos de línea. Esta vez no sé si están bien definidos los tipos, pero siempre me presenta en pantalla línea contínua.

   *c*) Tampoco funciona la definición de grosores de trazo.

# Parte I

# Manual del usuario

# Capítulo 1
# La variable global `geoVarGlobal`

Aquí va la información acerca de la variable global de GEO.

## 1.1. Unidades angulares

## 1.2. Dominios geodésicos

El dominio de algunas variables geodésicas utilizadas en geo es:

- **Latitud geodésica:** $]-\frac{\pi}{2}\frac{\pi}{2}[$.
- **Longitud geodésica:** $]-\pi\pi]$.
- **Acimut:** $[02\pi[$.

# Capítulo 2
# Tratamiento de modelos de geoide

Aquí va la información relativa al tratamiento de modelos del geoide.

# Capítulo 3
# Herramientas de dibujo

En este capítulo se desarrollará la capacidad de GEO para generar dibujos en distintos formatos.

# Capítulo 4
# Generación automática de la documentación

Uno de los puntos fuertes de la biblioteca GEO es su sistema de autodocumentación. Mediante una escritura estructurada de la ayuda de las funciones en los ficheros fuente[1] es posible mantener actualizada en todo momento la documentación de la biblioteca.

## 4.1. Tipos de documentación

En cuanto al contenido de la documentación, existen 3 tipos de manuales:

- **Manual del usuario:** Comprende la ayuda del usuario (cómo utilizar las funciones de la biblioteca) y la del desarrollador (estructura lógica y guía de programación). También incluye apéndices referentes a la instalación de *GNU Octave* y otros temas de interés.
- **Ayuda de las funciones:** Es un documento que, como su nombre indica, contiene la ayuda de todas las funciones de GEO.
- **Manual de referencia:** Este manual es una fusión de los dos anteriores. Constituye la ayuda más completa de GEO, por lo que es el documento que acompaña a la distribución de la biblioteca.

En cuanto al formato de los ficheros que componen la ayuda, también existen 3 tipos: DVI, PDF y HTML.

## 4.2. Generación de los ficheros fuente

El proceso de generación de los ficheros fuente de la documentación de GEO se lleva a cabo (dependiendo del tipo de manual a generar) mediante las funciones `geo_GeneraAyudaUsuarioLaTeX.m`, `geo_GeneraAyudaFuncionesLaTeX.m` o `geo_GeneraAyudaLaTeX.m`. La ejecución de cada una de ellas creará en el directorio de trabajo una carpeta que contendrá todos los ficheros necesarios para la correcta compilación con LaTeX del manual correspondiente. Estas funciones reciben como argumento una cadena de texto que contiene el nombre del usuario, para que sea imprimido en la portada del documento. Dicha cadena estará escrita con la misma sintaxis que si estuviésemos escribiendo directamente en un documento LaTeX. También ha de ser tenido en cuenta el modo en que *GNU Octave* o *MatLab* tratan determinados caracteres especiales cuando son introducidos como cadena de texto. De este modo, si, por ejemplo, queremos escribir la cadena `José`, que en LaTeX se escribe `Jos\'e`, habremos de teclear `Jos\''e` en *MatLab* y en versiones nuevas de *GNU Octave*, y `Jos\\''e` en versiones antiguas de *GNU Octave*.

[1] La documentación de las funciones en formato GEO se explica en el capítulo 6

## 4.3. Compilación de los ficheros fuente

Para compilar los ficheros fuente de la ayuda hemos de colocarnos en el directorio `compilar` de la carpeta creada por las funciones anteriormente señaladas.

### 4.3.1. Compilación automática

### 4.3.2. Compilación manual

La generación de los ficheros `*.dvi` o `*.pdf` se lleva a cabo en 4 etapas:

- **Primera pasada de compilación:** Se ejecuta el comando `latex` sobre el archivo global `ayuda`*`NombreAyuda`*`Geo.tex`[2] dos veces (`latex ayuda`*`NombreAyuda`*`Geo.tex`).
- **Generación de los índices:** Para generar los distintos índices alfabéticos se utiliza la aplicación `makeindex` de la distribución de TEX correspondiente. También nos apoyaremos en los archivos de estilo `estiloIndGeo.ist` y `estiloIndCitasGeo.ist`, que personalizan la apariencia de los índices. En el manual de referencia en formato PDF existen 5 tipos de índice alfabético: índice terminológico, de funciones, de programas de prueba, de ficheros y de citas bibliográficas. En el caso del manual en formato HTML sólo existe el índice alfabético. Los archivos que serán procesados por `makeindex` tienen los siguientes nombres (en el mismo orden que los índices citados): `ter.idx`, `fun.idx`, `pru.idx`, `fich.idx` y `cit.idx`.
- **Generación de la bibliografía:** Se ejecuta el comando `bibtex` sobre el archivo auxiliar `ayuda`*`NombreAyuda`*`Geo.aux` para generar el archivo de referencias bibliográficas correspondiente (`bibtex ayuda`*`NombreAyuda`*`Geo.aux`).
- **Segunda pasada de compilación:** Se repite la secuencia de comandos de la primera pasada de compilación. De esta forma quedan resueltas todas las referencias internas del documento, los índices y la bibliografía. Si, además del documento `*.dvi`, queremos generar un `*.pdf`, tenemos dos opciones:
  - Convertir el archivo DVI creado a PDF. Para ello se ejecuta el comando `dvipdfm` sobre el archivo `ayuda`*`NombreAyuda`*`Geo.dvi` (`dvipdfm ayuda`*`NombreAyuda`*`Geo.dvi`). Esto generará el archivo `ayuda`*`NombreAyuda`*`Geo.pdf`, aunque se perderán todos los hipervínculos de los índices, la bibliografía, etc.
  - Ejecutar el comando `pdflatex` sobre el archivo `ayuda`*`NombreAyuda`*`Geo.tex` (`pdflatex ayuda`*`NombreAyuda`*`Geo.tex`). Esto generará el archivo `ayuda`*`NombreAyuda`*`Geo.pdf`.

[2] El nombre del archivo global de compilación depende del tipo de ayuda generado y puede ser `ayudaUsuarioGeo.tex`, `ayudaFuncionesGeo.tex` o `ayudaGeo.tex`, dependiendo de si se ha generado el manual del usuario, la ayuda de las funciones o el manual de referencia, respectivamente.

# Parte II

# Manual del desarrollador

Capítulo 5

# Cómo escribir código fuente para GEO

En este capítulo se explica la sintaxis que debe seguir cualquier código fuente que quiera formar parte de GEO.

## 5.1. Cómo escribir funciones en lenguaje *GNU Octave*

## 5.2. Cómo escribir funciones en lenguaje *C/C++*

## 5.3. Cómo escribir código *C/C++* para la biblioteca `libgeo.a`

# Capítulo 6
# Documentación de la biblioteca GEO

En este capítulo se explica cómo generar la documentación de GEO.

# Capítulo 7
# Propagación de errores en GEO

En las primeras versiones de GEO, cuando en una función era necesario calcular la propagación de errores esta tarea se indicaba internamente con una variable de la siguiente manera:

```
controlVarianza = 0;
if nargin==numeroMaximo
    esEscalarVCA = geo_EsEscalar(varCovA);
    esEscalarVCB = geo_EsEscalar(varCovB);
    if (esEscalarVCA==1)&(esEscalarVCB==1)
        if (varCovA==-1)&(varCovB==-1)
            controlVarianza = 1;
        end
    end
else
    controlVarianza = 1;
end
```

De esta manera, cuando la variable `controlVarianza` valía 0 quería decir que se iba a realizar la propagación de errores y, cuando valía 1 se indicaba que no se realizaría. Además, como se puede ver en el trozo de código anterior (que es un ejemplo genérico), las comprobaciones del valor de los argumentos correspondientes a las matrices de varianza-covarianza de las observaciones se hacían explícitamente en el código fuente de la función. En la actualidad, este modo de proceder ha sido desechado y se han introducido los siguientes cambios:

1. Se han cambiado los valores de la variable de control, pasando a tomar los indicadores 1 cuando se vaya a realizar propagación y 0 cuando no se vaya a calcular. También se ha cambiado el nombre de dicha variable, pasando de `controlVarianza` a `ctrlVar`.

2. La comprobación del cálculo de la propagación se realiza ahora mediante la llamada a una sola función, lo que tiene las ventajas de hacer el fichero fuente más legible y ahorrar líneas de código.

La nueva función se llama `geo_HayQueCalcularErrores.m`, y se encarga de hacer las comprobaciones necesarias con las matrices varianza covarianza pasadas internamente. El trozo de código mostrado anteriormente tendrá el siguiente aspecto al utilizar la función citada:

```
if nargin==numeroMaximo
    ctrlVar = geo_HayQueCalcularErrores(varCovA,varCovB);
else
    ctrlVar = 0;
end
```

# Capítulo 8
# Optimización de la velocidad de ejecución del código

He aquí algunos consejos para mejorar la velocidad de ejecución de los programas escritos en *GNU Octave*:

- Use iteración en lugar de recursión siempre que sea posible. Las llamadas a funciones son lentas en *GNU Octave*.
- Evite redimensionar matrices innecesariamente. Cuando construya una matriz de resultados a partir de una serie de cálculos, asigne desde el principio el tamaño final de la matriz e introduzca seguidamente los datos necesarios. Escriba, por ejemplo:

```
resultado = zeros(fil,col);
for i=inicio:fin
    r1 = ...;
    r2 = ...;
    resultado(r1,r2) = nuevoValor;
end
```

  en lugar de:

```
resultado = [];
for i=inicio:fin
    resultado = [resultado,nuevoValor];
end
```

- Evite utilizar las funciones `eval` y `feval`, ya que requieren que *GNU Octave* analice la entrada o busque el nombre de la función en la tabla de símbolos.

# Parte III

# Ayuda de las funciones

# Capítulo 9
# Sección 01-global

En esta carpeta se almacena la función que contiene la variable global de toda la biblioteca.

- **geo_VarGlobal.m**:

```
Funcion:    geo_VarGlobal.m

Proposito:  Define las variables globales para la biblioteca

Entradas:   Nada

Salidas:    Define en memoria la estructura geoVarGlobal

Ejemplo:    geo_VarGlobal;

Usa geoVarGlobal: Si

Original:   23-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
            09-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Adicion de sentencias de test
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reestructuracion de los ficheros de datos y de la construccion
              de sus rutas de acceso
            11-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Adicion de variables para el trabajo con funciones que definen
              su ayuda a traves de sentencias #define
            10-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Adicion de directorios para los datos binarios de GEO fuera de
              la carpeta principal
```

# Capítulo 10
# Sección 02-general

En esta carpeta se almacenan funciones de utilidad general.

## 10.1. Carpeta 01-controlTipoDato

En esta carpeta se almacenan las funciones relacionadas con el control del tipo de datos.

- **geo_EsEntero.m**:

```
Funcion:    geo_EsEntero.m

Proposito:  Comprueba si el argumento pasado es un numero entero

Entradas:   - Una variable

Salidas:    - Si el argumento de entrada es un escalar:
              - 1, si es un numero entero
              - 0, en caso contrario
            - Si el argumento de entrada es un vector columna:
              - Un vector columna del mismo tamanyo que el vector de entrada
                que contiene en cada posicion correspondiente al vector de
                entrada:
                - 1, si en la posicion correspondiente hay un numero entero
                - 0, en caso contrario
            - Si el argumento de entrada no es ni un escalar ni un vector
              columna:
              - 0, sea cual sea el argumento

Ejemplo:    esEntero=geo_EsEntero(variable);

Usa geoVarGlobal: No

Original:   23-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 05-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsEscalar.m**:

```
Funcion:    geo_EsEscalar.m
```

```
Proposito:  Comprueba si el argumento pasado es un escalar

Entradas:   - Una variable

Salidas:    - 1, si el argumento de entrada es un escalar
            - 0, en caso contrario

Ejemplo:    esEscalar=geo_EsEscalar(variable);

Usa geoVarGlobal: No

Original:   23-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 05-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsFichero.m**:

```
Funcion:    geo_EsFichero.m

Proposito:  Comprueba si el argumento pasado es un identificador de fichero
            abierto

Entradas:   - Una variable

Salidas:    - 1, si el argumento pasado es un identificador de fichero abierto
            - 0, en caso contrario

Ejemplo:    esFichero=geo_EsFichero(variable);

Usa geoVarGlobal: No

Original:   23-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 05-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsMatriz.m**:

```
Funcion:    geo_EsMatriz.m

Proposito:  Comprueba si el argumento pasado es una matriz (para Octave y
            MatLab un escalar tambien es una matriz)

Entradas:   - Una variable

Salidas:    - 1, si el argumento correspondiente es una matriz
            - 0, en caso contrario

Ejemplo:    esMatriz=geo_EsMatriz(variable);
```

```
Usa geoVarGlobal: No

Original:   23-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 05-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsMatrizSimetrica.m**:

```
Funcion:    geo_EsMatrizSimetrica.m

Proposito:  Comprueba si el argumento pasado es una matriz simetrica

Entradas:   - Una matriz
            Argumentos opcionales:
              - Tolerancia para evitar problemas numericos. Si este argumento
                se omite, se coge como tolerancia el valor epsilon de la
                maquina (funcion 'eps')

Salidas:    - Dimensiones de la matriz si el argumento correspondiente es una
              matriz simetrica
            - 0, en caso contrario

Ejemplo:    esSimetrica=geo_EsMatrizSimetrica(matriz);
            esSimetrica=geo_EsMatrizSimetrica(matriz,tolerancia);

Usa geoVarGlobal: Si

Nota 1:     Esta funcion se ha programado para proporcionar compatibilidad
            entre GNU Octave y MatLab, ya que MatLab no dispone de la funcion
            issymmetric(). El codigo es una adaptacion para GEO (la parte de
            calculo es la misma) del que ejecuta la funcion issymmetric() de
            GNU Octave. Los creditos originales de la funcion citada son:
            Author: A. S. Hodel <scotte@eng.auburn.edu>
            Created: August 1993
            Adapted-By: jwe

Nota 2:     Una matriz X se considera simetrica si:
            norm(X-X',inf)/norm(X,inf)<=tolerancia
            Ver el codigo de la funcion issymmetric() de GNU Octave

Nota 3:     En la ayuda de la funcion issymmetric() de GNU Octave, una matriz
            X se considera simetrica si:
            norm(X-X',inf)/norm(X,inf)<tolerancia,
            pero en el codigo de la funcion se considera simetrica si:
            norm(X-X',inf)/norm(X,inf)<=tolerancia,
            por lo que en la programacion de esta funcion se ha seguido este
            ultimo criterio

Original:   17-03-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 05-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
```

```
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsNatural.m**:

```
Funcion:    geo_EsNatural.m

Proposito:  Comprueba si el argumento pasado es un numero natural. Se
            considera el conjunto de los numeros naturales como el de los
            enteros positivos N = {1,2,3,...}

Entradas:   - Una variable

Salidas:    - Si el argumento de entrada es un escalar:
              - 1, si es un numero natural
              - 0, en caso contrario
            - Si el argumento de entrada es un vector columna:
              - Un vector columna del mismo tamanyo que el vector de entrada
                que contiene en cada posicion correspondiente al vector de
                entrada:
                - 1, si en la posicion correspondiente hay un numero natural
                - 0, en caso contrario
            - Si el argumento de entrada no es ni un escalar ni un vector
              columna:
              - 0, sea cual sea el argumento

Ejemplo:    esNatural=geo_EsNatural(variable);

Usa geoVarGlobal: No

Original:   23-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 05-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsTexto.m**:

```
Funcion:    geo_EsTexto.m

Proposito:  Comprueba si el argumento pasado es una cadena de texto

Entradas:   - Una variable

Salidas:    - 1, si el argumento de entrada es una cadena de texto
            - 0, en caso contrario

Ejemplo:    esTexto=geo_EsTexto(variable);

Usa geoVarGlobal: No

Nota:       Esta funcion ha sido programada para proporcionar compatibilidad
            con versiones antiguas de GNU Octave

Original:   23-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
Revisiones: 05-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsVectorColumna.m**:

```
Funcion:    geo_EsVectorColumna.m

Proposito:  Comprueba si el argumento pasado es un vector columna

Entradas:   - Una variable

Salidas:    - 1, si el argumento correspondiente es un vector columna
            - 0, en caso contrario

Ejemplo:    esVectorColumna=geo_EsVectorColumna(variable);

Usa geoVarGlobal: No

Original:   12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geo_TipoDato.m**:

```
Funcion:    geo_TipoDato.m

Proposito:  Determina el tipo de una variable entre los siguientes:
            - geoLista
            - escalar
            - matriz
            - cadenaDeTexto
            - estructura
            - <tipoDesconocido>

Entradas:   Dos posibilidades:
            - Una variable
            - Sin argumento de entrada, la funcion devuelve una geoLista que
              contiene todos los tipos de dato reconocidos, excepto el tipo
              'fichero'

Salidas:    - El tipo del argumento de entrada, como cadena de texto

Ejemplo:    tipo=geo_TipoDato('geodesia') -> tipo='cadenaDeTexto'

Usa geoVarGlobal: No

Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 05-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

## 10.2. Carpeta 02-controlDominio

En esta carpeta se almacenan las funciones relacionadas con el control de dominio de variables geodésicas.

- **geo_ControlDominio.m**:

```
Funcion:    geo_ControlDominio.m

Proposito:  Comprueba si la variable de entrada esta en el dominio correcto,
            de acuerdo a los limites definidos en geoVarGlobal.var(i)

Entradas:   - Variable a comprobar (escalar o vector columna), en las mismas
              unidades que en geoVarGlobal.var(i).unidadesNaturales
            - Nombre de la variable, correspondiente al campo
              geoVarGlobal.var(i).nombre

Salidas:    - 1, si la variable o variables estan en el dominio correcto
            - Un error en caso contrario

Ejemplo:    correcto=geo_ControlDominio(variable,'acimut');

Usa geoVarGlobal: Si

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_ControlDominioAF.m**:

```
Funcion:    geo_ControlDominioAF.m

Proposito:  Comprueba si las variables semieje mayor del elipsoide y
            aplanamiento estan en el dominio correcto, de acuerdo a los
            limites definidos en:
            geoVarGlobal.var(i).nombre = 'SEMIEJE_MAYOR_ELIPSOIDE' y
            geoVarGlobal.var(i).nombre = 'APLANAMIENTO_ELIPSOIDE'

Entradas:   - Semieje mayor del elipsoide (escalar o vector columna), en las
              mismas unidades que en geoVarGlobal.var(i).unidadesNaturales
            - Aplanamiento del elipsoide (escalar o vector columna), en las
              mismas unidades que en geoVarGlobal.var(i).unidadesNaturales

Salidas:    - 1, si las variables estan en el dominio correcto
            - Un error en caso contrario

Ejemplo:    correcto=geo_ControlDominioAF(a,f);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
Reorganizacion de la ayuda
Reescritura de la funcion para que no utilice la variable
varargin como contenedor de los argumentos de entrada
Adicion de sentencias de test
```

- **geo_ControlDominioDiaMes.m**:

```
Funcion:    geo_ControlDominioDiaMes.m

Proposito:  Comprueba si las variables dia y mes estan en el dominio correcto,
            de acuerdo a los limites definidos en:
            geoVarGlobal.var(i).nombre = 'MES' (los dias de cada mes se
            definen en esta misma funcion)

Entradas:   - Dia del mes (escalar o vector columna)
            - Mes del anyo (escalar o vector columna), en las mismas unidades
              que en geoVarGlobal.var(i).unidadesNaturales
            - Anyo (escalar o vector columna)

Salidas:    - 1, si las variables estan en el dominio correcto
            - Un error en caso contrario

Ejemplo:    correcto=geo_ControlDominioDiaMes(dia,mes,anyo);

Usa geoVarGlobal: No

Original:   03-03-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geo_ControlDominioInterpMalla.m**:

```
Funcion:    geo_ControlDominioInterpMalla.m

Proposito:  Comprueba si las coordenadas de los puntos a interpolar en una
            malla estan en el dominio correcto

Entradas:   - Latitud geodesica (en radianes) o coordenada Y del punto de
              trabajo (escalar o vector columna)
            - Longitud geodesica (en radianes) o coordenada X del punto de
              trabajo (escalar o vector columna)
            - Codigo identificador del sistema de coordenadas en que vienen
              expresados los puntos dato:
              0/1->coordenadas cartesianas/coordenadas geodesicas
            - Limite norte de la malla de datos: si el sistema de coordenadas
              es geodesico, latitud en radianes; si es cartesiano, maximo
              valor de la coordenada Y
            - Limite sur de la malla de datos: si el sistema de coordenadas es
              geodesico, latitud en radianes; si es cartesiano, minimo valor
              de la coordenada Y
            - Limite este de la malla de datos: si el sistema de coordenadas
              es geodesico, longitud en radianes; si es cartesiano, maximo
              valor de la coordenada X
            - Limite oeste de la malla de datos: si el sistema de coordenadas
              es geodesico, longitud en radianes; si es cartesiano, minimo
              valor de la coordenada X
            - Indicador de cobertura global en longitud: 0/1->no/si
```

```
              Si el sistema de coordenadas es cartesiano este argumento no
              sera tenido en cuenta en la ejecucion de la funcion

Salidas:    - 1, si las variables estan en el dominio correcto
            - Un error en caso contrario

Ejemplo:    [correcto]=geo_ControlDominioInterpMalla(latGeo,lonGeo,sistCoor,
            limNorte,limSur,limEste,limOeste,cobGlobLon);

Usa geoVarGlobal: Si

Nota:       Esta funcion no comprueba si el tipo de los datos pasados es
            correcto ni si estos tienen las dimensiones adecuadas. Tampoco
            comprueba si los datos referidos a la malla son correctos

Original:   04-03-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_ControlDominioLatGeoAF.m**:

```
Funcion:    geo_ControlDominioLatGeoAF.m

Proposito:  Comprueba si las variables latitud geodesica, semieje mayor del
            elipsoide y aplanamiento estan en el dominio correcto, de acuerdo
            a los limites definidos en:
            geoVarGlobal.var(i).nombre = 'LATITUD_GEODESICA',
            geoVarGlobal.var(i).nombre = 'SEMIEJE_MAYOR_ELIPSOIDE' y
            geoVarGlobal.var(i).nombre = 'APLANAMIENTO_ELIPSOIDE'

Entradas:   - Latitud geodesica (escalar o vector columna), en las mismas
              unidades que en geoVarGlobal.var(i).unidadesNaturales
            - Semieje mayor del elipsoide (escalar o vector columna), en las
              mismas unidades que en geoVarGlobal.var(i).unidadesNaturales
            - Aplanamiento del elipsoide (escalar o vector columna), en las
              mismas unidades que en geoVarGlobal.var(i).unidadesNaturales

Salidas:    - 1, si las variables estan en el dominio correcto
            - Un error en caso contrario

Ejemplo:    correcto=geo_ControlDominioLatGeoAF(latitudGeodesica,a,f);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_ControlDominioLatGeoLonGeo.m**:

```
Funcion:    geo_ControlDominioLatGeoLonGeo.m

Proposito:  Comprueba si las variables latitud geodesica, y longitud geodesica
            estan en el dominio correcto, de acuerdo a los limites definidos
            en:
            geoVarGlobal.var(i).nombre = 'LATITUD_GEODESICA' y
            geoVarGlobal.var(i).nombre = 'LONGITUD_GEODESICA'

Entradas:   - Latitud geodesica (escalar o vector columna), en las mismas
              unidades que en geoVarGlobal.var(i).unidadesNaturales
            - Longitud geodesica (escalar o vector columna), en las mismas
              unidades que en geoVarGlobal.var(i).unidadesNaturales

Salidas:    - 1, si las variables estan en el dominio correcto
            - Un error en caso contrario

Ejemplo:    correcto=geo_ControlDominioLatGeoLonGeo(latitud,longitud);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_ControlDominioLatGeoLonGeoHElip.m**:

```
Funcion:    geo_ControlDominioLatGeoLonGeoHElip.m

Proposito:  Comprueba si las variables latitud geodesica, longitud geodesica y
            altitud elipsoidal estan en el dominio correcto, de acuerdo a los
            limites definidos en:
            geoVarGlobal.var(i).nombre = 'LATITUD_GEODESICA',
            geoVarGlobal.var(i).nombre = 'LONGITUD_GEODESICA' y
            geoVarGlobal.var(i).nombre = 'ALTITUD_ELIPSOIDAL'

Entradas:   - Latitud geodesica (escalar o vector columna), en las mismas
              unidades que en geoVarGlobal.var(i).unidadesNaturales
            - Longitud geodesica (escalar o vector columna), en las mismas
              unidades que en geoVarGlobal.var(i).unidadesNaturales
            - Altitud elipsoidal (escalar o vector columna), en las mismas
              unidades que en geoVarGlobal.var(i).unidadesNaturales

Salidas:    - 1, si las variables estan en el dominio correcto
            - Un error en caso contrario

Ejemplo:    correcto=geo_ControlDominioLatGeoLonGeoHElip(latitud,longitud,h);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
```

```
                varargin como contenedor de los argumentos de entrada
                Adicion de sentencias de test
```

## 10.3. Carpeta 03-angulos

En esta carpeta se almacenan las funciones relacionadas con la conversión entre sistemas angulares.

- **geo_CentGdec.m**:

```
Funcion:    geo_CentGdec.m

Proposito:  Transforma un angulo de grados centesimales a grados sexagesimales
            en formato decimal

Entradas:   - Angulo a transformar (escalar o vector columna), en formato
              centesimal

Salidas:    - Angulo en grados sexagesimales decimales (escalar o vector
              columna)

Ejemplo:    gdec=geo_CentGdec(cent);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_CentGms.m**:

```
Funcion:    geo_CentGms.m

Proposito:  Transforma un angulo de grados centesimales a grados sexagesimales
            en formato grados-minutos-segundos (gms). En gms, 40.245689
            equivale a 40 grados, 24 minutos, 56.89 segundos

Entradas:   - Angulo a transformar (escalar o vector columna), en grados
              centesimales

Salidas:    - Angulo en grados sexagesimales formato grados-minutos-segundos
              (escalar o vector columna)

Ejemplo:    gms=geo_CentGms(cent);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_CentRad.m**:

```
Funcion:    geo_CentRad.m

Proposito:  Transforma un angulo de grados centesimales a radianes

Entradas:   - Angulo a transformar (escalar o vector columna), en formato
              centesimal

Salidas:    - Angulo en radianes (escalar o vector columna)

Ejemplo:    rad=geo_CentRad(cent);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_GdecCent.m**:

```
Funcion:    geo_GdecCent.m

Proposito:  Transforma un angulo de grados sexagesimales en formato decimal a
            grados centesimales

Entradas:   - Angulo a transformar (escalar o vector columna), en grados
              sexagesimales formato decimal

Salidas:    - Angulo en grados centesimales (escalar o vector columna)

Ejemplo:    cent=geo_GdecCent(gdec);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_GdecGms.m**:

```
Funcion:    geo_GdecGms.m

Proposito:  Transforma un angulo de grados sexagesimales en formato decimal a
            grados sexagesimales en formato grados-minutos-segundos (gms)
            En gms, 40.245689 equivale a 40 grados, 24 minutos, 56.89 segundos

Entradas:   - Angulo a transformar (escalar o vector columna), en grados
              sexagesimales formato decimal
```

```
Salidas:    - Angulo en grados sexagesimales formato grados-minutos-segundos
              (escalar o vector columna)

Ejemplo:    gms=geo_GdecGms(gdec);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_GdecRad.m**:

```
Funcion:    geo_GdecRad.m

Proposito:  Transforma un angulo de grados sexagesimales en formato decimal a
            radianes

Entradas:   - Angulo a transformar (escalar o vector columna), en grados
              sexagesimales formato decimal

Salidas:    - Angulo en radianes (escalar o vector columna)

Ejemplo:    rad=geo_GdecRad(gdec);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_GmsCent.m**:

```
Funcion:    geo_GmsCent.m

Proposito:  Transforma un angulo de grados sexagesimales en formato grados-
            minutos-segundos (gms) a grados centesimales. En gms, 40.245689
            equivale a 40 grados, 24 minutos, 56.89 segundos

Entradas:   - Angulo a transformar (escalar o vector columna), en formato
              grados-minutos-segundos

Salidas:    - Angulo en grados centesimales (escalar o vector columna)

Ejemplo:    cent=geo_GmsCent(gms);

Usa geoVarGlobal: No
```

```
Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_GmsGdec.m**:

```
Funcion:    geo_GmsGdec.m

Proposito:  Transforma un angulo de grados sexagesimales en formato grados-
            minutos-segundos (gms) a grados sexagesimales en formato decimal
            En gms, 40.245689 equivale a 40 grados, 24 minutos, 56.89 segundos

Entradas:   - Angulo a transformar (escalar o vector columna), en formato
              grados-minutos-segundos

Salidas:    - Angulo en grados sexagesimales formato decimal (escalar o vector
              columna)

Ejemplo:    gdec=geo_GmsGdec(gms);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_GmsRad.m**:

```
Funcion:    geo_GmsRad.m

Proposito:  Transforma un angulo de grados sexagesimales en formato grados-
            minutos-segundos (gms) a radianes. En gms, 40.245689 equivale a
            40 grados, 24 minutos, 56.89 segundos

Entradas:   - Angulo a transformar (escalar o vector columna), en formato
              grados-minutos-segundos

Salidas:    - Angulo en radianes (escalar o vector columna)

Ejemplo:    rad=geo_GmsRad(gms);

Usa geoVarGlobal: Si

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-08-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Eliminado el uso de la funcion geo_Find en algunas sentencias
              para optimizar la velocidad de ejecucion
            12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_RadCent.m**:

```
Funcion:    geo_RadCent.m

Proposito:  Transforma un angulo de radianes a grados centesimales

Entradas:   - Angulo a transformar (escalar o vector columna), en radianes

Salidas:    - Angulo en grados centesimales (escalar o vector columna)

Ejemplo:    cent=geo_RadCent(rad);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_RadGdec.m**:

```
Funcion:    geo_RadGdec.m

Proposito:  Transforma un angulo de radianes a grados sexagesimales en
            formato decimal

Entradas:   - Angulo a transformar (escalar o vector columna), en radianes

Salidas:    - Angulo en grados sexagesimales formato decimal (escalar o vector
              columna)

Ejemplo:    gdec=geo_RadGdec(rad);

Usa geoVarGlobal: No

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_RadGms.m**:

```
Funcion:    geo_RadGms.m

Proposito:  Transforma un angulo de radianes a grados sexagesimales en formato
            grados-minutos-segundos (gms). En gms, 40.245689 equivale a
```

```
            40 grados, 24 minutos, 56.89 segundos

Entradas:   - Angulo a transformar (escalar o vector columna), en radianes

Salidas:    - Angulo en grados sexagesimales formato grados-minutos-segundos
              (escalar o vector columna)

Ejemplo:    gms=geo_RadGms(rad);

Usa geoVarGlobal: Si

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-08-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Eliminado el uso de la funcion geo_Find en algunas sentencias
              para optimizar la velocidad de ejecucion
            12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

## 10.4. Carpeta 04-geoLista

En esta carpeta se almacenan las funciones relacionadas con el trabajo con listas.

- **geo_DameElementoLista.m**:

```
Funcion:    geo_DameElementoLista.m

Proposito:  Extrae el enesimo elemento de una geoLista

Entradas:   - Una geoLista
            - Posicion del elemento a extraer, un numero natural

Salidas:    - El elemento de la posicion seleccionada

Ejemplo:    elemento=geo_DameElementoLista(geoLista,3);

Usa geoVarGlobal: Si

Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsLista.m**:

```
Funcion:    geo_EsLista.m

Proposito:  Determina si el argumento pasado es una geoLista

Entradas:   - Una geoLista
```

```
Salidas:    - 1, si el argumento de entrada es una geoLista
            - 0, en caso contrario

Ejemplo:    es=geo_EsLista(lista);

Usa geoVarGlobal: No

Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_ExisteEnLista.m**:

```
Funcion:    geo_ExisteEnLista.m

Proposito:  Comprueba si el elemento pasado existe como elemento en la
            geoLista Pasada

Entradas:   - Una variable
            - Una geoLista

Salidas:    - Vector columna con la posicion en la geoLista donde existan
              elementos iguales al pasado. Si la geoLista no contiene ningun
              elemento igual se devuelve el valor 0
            - Numero de elementos iguales encontrados

Ejemplo:    existe=geo_ExisteEnLista(variable,geoLista);

Usa geoVarGlobal: Si

Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_HayRepeticionesEnLista.m**:

```
Funcion:    geo_HayRepeticionesEnLista.m

Proposito:  Comprueba si hay al menos dos elementos iguales en una geoLista

Entradas:   - Una geoLista

Salidas:    - Posicion el la geoLista del primer elemento igual, numero entero
            - Posicion del segundo elemento igual, numero entero
            Si todos los elementos son distintos las variables anteriores
            valen cero

Ejemplo:    [primero,segundo]=geo_HayRepeticionesEnLista(geoLista);
```

```
Usa geoVarGlobal: Si

Original:   22-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Corregido bug que hacia que los controles no se restaurasen si
              se encontraban dos elementos iguales
              Adicion de sentencias de test
```

- **geo_InvierteLista.m**:

```
Funcion:    geo_InvierteLista.m

Proposito:  Invierte el orden de los elementos de una geoLista

Entradas:   - Una geoLista

Salidas:    - Una nueva geoLista creada con los elementos de la geoLista de
              entrada en orden inverso

Ejemplo:    nuevaLista=geo_InvierteLista(geoLista);

Usa geoVarGlobal: Si

Nota:       Esta funcion no hace un uso eficiente de la memoria en el sentido
            de que no declara el tamany de un cell array antes de utilizarlo
            cuando su tamanyo es conocido. Hay que averiguar si las versiones
            antiguas de GNU Octave y MatLab proveen la funcion necesaria para
            declarar un cell array de un tamanyo determinado (en GNU Octave
            3.0.3 la funcion es cell()) y modificar el codigo en consecuencia

Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_Lista.m**:

```
Funcion:    geo_Lista.m

Proposito:  Crea una geoLista con los elementos pasados como argumentos

Entradas:   - Elementos que contendra la lista

Salidas:    - Lista con los argumentos de entrada

Ejemplo:    lista=geo_Lista(1,'a',4,[1 2]);

Usa geoVarGlobal: No

Nota:       Esta funcion no hace un uso eficiente de la memoria en el sentido
```

```
              de que no declara el tamany de un cell array antes de utilizarlo
              cuando su tamanyo es conocido. Hay que averiguar si las versiones
              antiguas de GNU Octave y MatLab proveen la funcion necesaria para
              declarar un cell array de un tamanyo determinado (en GNU Octave
              3.0.3 la funcion es cell()) y modificar el codigo en consecuencia

  Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Reorganizacion de la ayuda
                Adicion de sentencias de test
```

- **geo_LongitudLista.m**:

```
  Funcion:    geo_LongitudLista.m

  Proposito:  Calcula el numero de elementos de una geoLista

  Entradas:   - Una geoLista

  Salidas:    - Numero de elementos de la geoLista, excluido el identificador

  Ejemplo:    numero=geo_LongitudLista(geoLista);

  Usa geoVarGlobal: Si

  Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Reorganizacion de la ayuda
                Reescritura de la funcion para que no utilice la variable
                varargin como contenedor de los argumentos de entrada
                Adicion de sentencias de test
```

- **geo_RepiteEnLista.m**:

```
  Funcion:    geo_RepiteEnLista.m

  Proposito:  Anade un elemento multiples veces al final de una geoLista

  Entradas:   - Una geoLista
              - Elemento a anadir. Si es una geoLista, sus elementos se anaden
                individualmente
              - Numero de veces que se repetira el elemento a anadir

  Salidas:    - Una nueva geoLista con todos los elementos

  Ejemplo:    a=geo_Lista(1,2);
              nuevaLista=geo_RepiteEnLista(a,'hola',2) -> {1,2,hola,hola}

  Usa geoVarGlobal: Si

  Nota:       Esta funcion no hace un uso eficiente de la memoria en el sentido
              de que no declara el tamany de un cell array antes de utilizarlo
              cuando su tamanyo es conocido. Hay que averiguar si las versiones
              antiguas de GNU Octave y MatLab proveen la funcion necesaria para
              declarar un cell array de un tamanyo determinado (en GNU Octave
```

```
            3.0.3 la funcion es cell()) y modificar el codigo en consecuencia

Original:   02-09-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_UneALista.m**:

```
Funcion:    geo_UneALista.m

Proposito:  Anade elementos al final de una geoLista

Entradas:   - Una geoLista
            Elementos opcionales
              - Elementos a anadir. Si alguno de ellos es una geoLista, sus
                elementos se anyaden individualmente

Salidas:    - Una nueva geoLista con todos los elementos
            Si solo se pasa un argumento de entrada, la salida sera esa misma
            geoLista

Ejemplo:    a=geo_Lista(1,2);
            b=geo_Lista(3,[1 2]);
            c='geodesia'
            nuevaLista=geo_UneALista(a,b,c) -> {1,2,3,[1 2],'geodesia'}

Usa geoVarGlobal: Si

Nota:       Esta funcion no hace un uso eficiente de la memoria en el sentido
            de que no declara el tamany de un cell array antes de utilizarlo
            cuando su tamanyo es conocido. Hay que averiguar si las versiones
            antiguas de GNU Octave y MatLab proveen la funcion necesaria para
            declarar un cell array de un tamanyo determinado (en GNU Octave
            3.0.3 la funcion es cell()) y modificar el codigo en consecuencia

Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Modificacion de la funcion para que funcione con un solo
              argumento de entrada
              Adicion de sentencias de test
```

## 10.5. Carpeta 05-matrices

En esta carpeta se almacenan las funciones relacionadas con el trabajo con matrices.

### 10.5.1. Carpeta 01-matricesGeneral

En esta carpeta se almacenan las funciones relacionadas con el trabajo con matrices generales.

- **geo_MatrizVector.m**:

```
Funcion:    geo_MatrizVector.m

Proposito:  Transforma una matriz en un vector constituido por una sucesion de
            sus filas o sus columnas

Entradas:   - Matriz a transformar
            Argumentos opcionales:
              - Cadena de texto 'FILAS' o 'COLUMNAS', dependiendo de que el
                vector de salida este constituido, respectivamente, por las
                filas o las columnas de la matriz de entrada. Si se omite,
                este argumento vale 'FILAS'

Salidas:    - Vector columna formado por las filas o las columnas de la matriz
              de entrada

Ejemplo:    vector=geo_MatrizVector(matriz);
            vector=geo_MatrizVector(matriz,'COLUMNAS');

Usa geoVarGlobal: Si

Original:   14-12-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Reescritura del codigo para que utilice la funcion reshape()
              cuando el vector de salida esta formado por las columnas de la
              matriz de entrada
              Adicion de sentencias de test
            19-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              La comprobacion del segundo argumento de entrada se hace ahora
              en el cuerpo de ejecucion de la funcion, aprovechando la
              estructura if...elseif...else...end de seleccion de dimension
```

- **geo_OrdenaMatriz.m**:

```
Funcion:    geo_OrdenaMatriz.m

Proposito:  Ordena un vector (fila o columna) o una matriz en orden ascendente
            o descendente. Si trabajamos con una matriz, ordena una fila o
            columna y el resto las ajusta segun la fila o columna ordenada

Entradas:   - Variable a ordenar
            - Cadena de texto 'MENOS-MAS', si queremos ordenar de menor a
              mayor, o 'MAS-MENOS', si queremos ordenar de mayor a menor
              Si la variable donde vamos a buscar es una matriz:
              - Cadena de texto 'fila' o 'columna', que indica la dimension
                donde vamos a ordenar
              - Numero de la fila o columna a ordenar

Salidas:    - La matriz de entrada con la fila o columna indicada ordenada en
              el orden elegido y con el resto de elementos ajustados segun la
              fila o columna ordenada. Si en la ordenacion se encuentran dos
              valores iguales no se altera su orden relativo en la matriz
              original

Ejemplo:    vectorOrdenado=geo_OrdenaMatriz(vector,'MENOS-MAS');
```

```
            matrizOrdenada=geo_OrdenaMatriz(matriz,'MENOS-MAS','COLUMNA',3);
            matrizOrdenada=geo_OrdenaMatriz(matriz,'MAS-MENOS','FILA',1);

  Usa geoVarGlobal: Si

  Original:   01-09-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: 14-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Reorganizacion de la ayuda
                Reescritura de la funcion para que no utilice la variable
                varargin como contenedor de los argumentos de entrada
                Reescritura de la funcion para que utilice la funcion sort() de
                las versiones nuevas de GNU Octave, es la que se permite
                seleccionar el orden y la dimension para realizar la ordenacion
                Se mantiene la compatibilidad con las versiones antiguas de
                GNU Octave
                Adicion de sentencias de test
              15-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Optimizacion del codigo para el trabajo con versiones antiguas
                de GNU Octave. Se eliminan los bucles for en la recuperacion de
                las posiciones de los elementos que no pertenecen a la fila
                ordenada de la matriz de trabajo
              19-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Se anyade comprobacion de cadena de texto para el segundo y
                tercer argumentos de entrada
```

- **geo_OrdenaMatrizPlantilla.m**:

```
  Funcion:    geo_OrdenaMatrizPlantilla.m

  Proposito:  Ordena un vector (fila o columna) o una matriz en el mismo orden
              que un vector (plantilla) pasado. Si trabajamos con una matriz,
              ordena una fila o columna y el resto las ajusta segun la fila o
              columna ordenada

  Entradas:   - Variable a ordenar
              - Cadena de texto 'FILA' o 'COLUMNA', que indica la dimension
                donde vamos a ordenar
              - Numero de la fila o columna a ordenar
              - Vector de referencia (plantilla)

  Salidas:    - La matriz de entrada con la fila o columna indicada ordenada en
                el mismo orden que los elementos de la plantilla que contenga y
                con el resto ajustados segun la fila o columna ordenada. Si en
                la ordenacion se encuentran dos valores iguales no se altera su
                orden relativo en la matriz original. Si en la matriz a ordenar
                hay elementos que no existen en el vector de referencia, estos
                se almacenan al final de la matriz ordenada en el mismo orden
                relativo de la matriz original

  Ejemplo:    matOrd=geo_OrdenaMatrizPlantilla(matriz,'FILA',3,plantilla);
              matOrd=geo_OrdenaMatrizPlantilla(matriz,'COLUMNA',4,plantilla);

  Usa geoVarGlobal: Si

  Nota 1:     Si la plantilla contiene algun elemento repetido y que, a su vez,
              existe en la dimension correspondiente de la matriz a ordenar, la
              matriz ordenada de salida sera mas grande que la matriz
```

```
            desordenada original. La razon es que, como en la plantilla hay
            elementos repetidos, se repetiran filas o columnas de la matriz
            original
  Nota 2:   Los elementos de la plantilla que no existan en la dimension
            correspondiente de la matriz original no aparecen en la matriz
            ordenada, por lo que las posiciones absolutas de los elementos de
            la plantilla y los de la matriz ordenada pueden no coincidir. Solo
            coincidiran cuando todos los elementos de la plantilla existan en
            la dimension correspondiente de la matriz original

  Original: 17-11-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: 15-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
            19-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Reescritura de la funcion para que la plantilla sea un vector,
              en lugar de una matriz (como era en versiones anteriores). Esto
              ahorra espacio en memoria y argumentos de entrada con respecto a
              las versiones anteriores de la funcion
              Adicion de sentencias de test
```

- **geo_PosGruposVector.m**:

```
  Funcion:    geo_PosGruposVector.m

  Proposito:  Calcula las posiciones en un vector columna de todos los elementos
              iguales entre si que esten situados de forma consecutiva

  Entradas:   - Un vector columna

  Salidas:    - Una matriz de tamano Nx3 que contiene el elemento del vector
                original y las posiciones en que comienza y termina un grupo de
                elementos iguales consecutivos de ese elemento. Su estructura es
                la siguiente:
                posiciones=[dato1  filInicio1  filFin1
                            ........................
                            datoN  filInicioN  filFinN];
                Cada fila corresponde a un grupo de elementos iguales:
                - Col. 1: Dato almacenado en el vector original
                - Col. 2: Fila del vector original en la que se encuentra el
                          dato de trabajo
                - Col. 3: Fila del vector original en la que termina una serie
                          de datos iguales al de trabajo

  Ejemplo:    posiciones=geo_PosGruposVector(vector);

  Usa geoVarGlobal: No

  Nota:       Veamos el funcionamiento con un ejemplo:
              El siguiente vector columna:
              vector = [2 5 5 2 2 2 8]';
              Produce la siguiente salida:
              posiciones = [2 1 1
                            5 2 3
                            2 4 6
                            8 7 7];
```

```
            Que ha de ser entendida como:
            - El dato 2 comienza en la fila 1 y termina en la fila 1 del
              vector, es decir, hay un dato 2 en la fila 1 del vector original
            - El dato 5 comienza en la fila 2 y termina en la fila 3 del
              vector, es decir, hay un dato 5 en las filas 2 y 3 del vector
              original
            - El dato 2 comienza en la fila 4 y termina en la fila 6 del
              vector, es decir, hay un dato 2 en las filas 4, 5 y 6 del vector
              original
            - El dato 8 comienza en la fila 7 y termina en la fila 7 del
              vector, es decir, hay un dato 2 en la fila 7 del vector original

Original:   15-08-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
            19-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_Redimensiona.m**:

```
Funcion:    geo_Redimensiona.m

Proposito:  Redimensiona una matriz manteniendo su contenido original

Entradas:   - Un escalar o una matriz
            - Numero de filas de la matriz redimensionada, como numero natural
            - Numero de columnas de la matriz redimensionada, como numero
              natural

Salidas:    - Matriz redimensionada que contiene a la matriz original en su
              cuadrante superior izquierdo (cada elemento de la matriz
              original esta en la misma posicion en la matriz redimensionada)
              Si las nuevas dimensiones pasadas son menores que las de la
              matriz original se devuelve la submatriz correspondiente

Ejemplo:    matrizRedim=geo_Redimensiona(matriz,filas,columnas);

Usa geoVarGlobal: Si

Nota 1:     Esta funcion ha sido programada para proporcionar compatibilidad
            entre GNU Octave y MatLab y entre diferentes versiones de Octave
            (las antiguas no disponen de la funcion resize())
Nota 2:     La funcion geo_Redimensiona() tiene menos funcionalidades que la
            funcion resize de MatLab y de las versiones recientes de
            GNU Octave

Original:   02-09-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
            19-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Corregido bug en la comprobacion del tipo del segundo y tercer
              argumentos de entrada
```

```
              Adicion de sentencias de test
```

- **geo_VectorMatriz.m**:

```
Funcion:    geo_VectorMatriz.m

Proposito:  Convierte un vector fila o columna en una matriz

Entradas:   - Vector fila o columna
            - Numero de filas de la matriz a construir
            - Numero de columnas de la matriz a construir
            Argumentos opcionales:
              - Cadena de texto 'FILAS' o 'COLUMNAS', dependiendo de que la
                matriz de salida se vaya rellenando, respectivamente, por
                filas o por columnas con los datos del vector. Si se omite,
                este argumento vale 'FILAS'

Salidas:    - Matriz formada a partir del vector de entrada

Ejemplo:    matriz=geo_VectorMatriz(vector,fil,col);
            matriz=geo_VectorMatriz(vector,fil,col,'COLUMNAS');

Usa geoVarGlobal: Si

Original:   14-12-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
            19-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

### 10.5.2. Carpeta 02-matricesVector

En esta carpeta se almacenan las funciones relacionadas con el trabajo con matrices en formato vector.

- **geo_MatrizDiagMatrizComp.m**:

```
Funcion:    geo_MatrizDiagMatrizComp.m

Proposito:  Pasa de una matriz diagonal en formato vector a una completa en
            este mismo formato

Entradas:   - Vector fila o matriz con la definicion de la matriz (cada fila
              si hemos pasado una matriz) diagonal en formato vector

Salidas:    - Vector fila o matriz con la definicion de la matriz (cada fila
              si hemos pasado una matriz) completa con los elementos fuera de
              la diagonal principal iguales a 0
              De
                              matriz=[1  2  3
                                      4  5  6
                                      7  8  9];
              definida como
                              vecDiag=[1  5  9];
```

```
              pasamos a
                              vecComp=[1  0  0  5  0  9];

Ejemplo:    matrizCompleta=geo_MatrizDiagMatrizComp(matrizDiagonal);

Usa geoVarGlobal: No

Original:   12-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 19-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            24-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Actualizacion de la ayuda
```

- **geo_MatrizSimVector.m**:

```
Funcion:    geo_MatrizSimVector.m

Proposito:  Transforma una matriz simetrica a formato vector para su manejo
            por las funciones de la biblioteca

Entradas:   - Matriz simetrica (no se controla internamente)
            Argumento opcional:
            - Cadenas de texto 'TODO' o 'DIAGONAL', si se quiere extraer toda
              la parte triangular superior o solo la diagonal principal. Por
              defecto, este argumento vale 'TODO'

Salidas:    - Vector fila que contiene la parte triangular superior de la
              matriz de entrada
              De
                          original=[a11  a12  ...  a1n
                                    a12  a22  ...  a2n
                                     .    .   ...   .
                                     .    .   ...   .
                                    a1n  a2n  ...  ann];
              se extrae
              'TODO'      vector=[a11 a12 ... a1n a22 ... a2n ... ann];
              o
              'DIAGONAL'  vector=[a11 a22 ... ann];

Ejemplo:    vector=geo_MatrizSimVector(matriz);
            vector=geo_MatrizSimVector(matriz,'TODO');
            vector=geo_MatrizSimVector(matriz,'DIAGONAL');

Usa geoVarGlobal: Si

Nota:       La funcion no controla internamente si la matriz pasada es
            simetrica, por lo que funcionara con cualquier matriz cuadrada
            Esta funcion esta pensada para el trabajo con matrices
            varianza-covarianza

Original:   29-08-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 22-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
```

```
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_MatrizSimVectorInterna.m**:

```
Funcion:    geo_MatrizSimVectorInterna.m

Proposito:  Transforma una matriz simetrica a formato vector para su manejo
            por las funciones de la biblioteca. Esta funcion no tiene ningun
            tipo de control interno de errores, a fin de hacer mas rapida su
            ejecucion

Entradas:   - Matriz simetrica (no se controla internamente)
            Argumento opcional:
            - Cadenas de texto 'TODO' o 'DIAGONAL', si se quiere extraer toda
              la parte triangular superior o solo la diagonal principal. Por
              defecto, este argumento vale 'TODO'

Salidas:    - Vector fila que contiene la parte triangular superior de la
              matriz de entrada
              De
                          original=[a11  a12  ...  a1n
                                    a12  a22  ...  a2n
                                     .    .   ...   .
                                     .    .   ...   .
                                    a1n  a2n  ...  ann];
              se extrae
              'TODO'      vector=[a11 a12 ... a1n a22 ... a2n ... ann];
              o
              'DIAGONAL'  vector=[a11 a22 ... ann];

Ejemplo:    vector=geo_MatrizSimVector(matriz);
            vector=geo_MatrizSimVector(matriz,'TODO');
            vector=geo_MatrizSimVector(matriz,'DIAGONAL');

Usa geoVarGlobal: No

Original:   29-08-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 23-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_UneMatricesDiagonal.m**:

```
Funcion:    geo_UneMatricesDiagonal.m

Proposito:  Fusiona varias matrices cuadradas en una sola

Entradas:   - Una serie de matrices cuadradas separadas por comas

Salidas:    - Matriz cuadrada, diagonal por bloques, cada uno de los cuales se
              corresponde a cada matriz de entrada
```

```
Ejemplo:    matriz=geo_UneMatricesDiagonal(matriz1,matriz2,matriz3);

Usa geoVarGlobal: Si

Original:   25-08-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 23-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

- **geo_UneMatrizVectorDiagonal.m**:

```
Funcion:    geo_UneMatrizVectorDiagonal.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anyade a una matriz cuadrada normal varias matrices definidas como
            vectores

Entradas:   - Una matriz cuadrada
            Argumentos opcionales:
              - Grupos de dos argumentos, de la siguiente forma:
                - Vector fila o matriz con la definicion de una matriz en
                  formato vector (cada una de sus filas en el caso de haber
                  pasado una matriz)
                - Numero indicador de la dimensiones de la matriz
                  correspondiente a la definicion anterior, un numero natural
                  Solo hace falta un valor porque la matriz pasada es cuadrada
                  Si el numero pasado coincide con el numero de elementos del
                  vector o las columnas de la matriz, se considera que la
                  matriz es diagonal

Salidas:    - Matriz cuadrada, diagonal por bloques, cada uno de los cuales se
              corresponde a cada matriz de entrada

Ejemplo:    matriz=geo_UneMatrizVectorDiagonal(matriz,matriz1,3,matriz2,5);

Usa geoVarGlobal: Si

Original:   22-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 23-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
            24-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Anyadida comprobacion de la congruencia entre el numero de
              columnas de las definiciones de matriz pasadas y el numero
              indicador de las dimensiones de la matriz definida como vector
```

- **geo_VectorMatrizSim.m**:

```
Funcion:    geo_VectorMatrizSim.m

Proposito:  Crea una matriz simetrica a partir de vectores fila con la
            descripcion de sus elementos

Entradas:   Grupos de dos argumentos, de la siguiente forma:
```

```
            - Vector fila o matriz con la definicion de la parte triangular
              superior de una matriz simetrica (cada una de sus filas en el
              caso de haber pasado una matriz)
            - Numero indicador de la dimensiones de la matriz correspondiente
              a la definicion anterior, un numero natural. Solo hace falta un
              valor porque la matriz es cuadrada. Si el numero pasado coincide
              con el numero de elementos del vector o las columnas de la
              matriz, se considera que la matriz es diagonal

Salidas:    - Matriz diagonal por bloques, cada uno de los cuales es la matriz
              que define cada elemento de definicion pasado
              De
                          matriz1=[1 2]; matriz2=[1 2 3;4 5 6]; y matriz3=7;
              al ejecutar
                          geo_VectorMatrizSim(matriz1,2,matriz2,2,matriz3,1);
              se obtiene
                          matriz=[1  0  0  0  0  0  0
                                  0  2  0  0  0  0  0
                                  0  0  1  2  0  0  0
                                  0  0  2  3  0  0  0
                                  0  0  0  0  4  5  0
                                  0  0  0  0  5  6  0
                                  0  0  0  0  0  0  7];

Ejemplo:    matriz=geo_VectorMatrizSim(matriz1,3,matriz2,5,matriz3,1);

Usa geoVarGlobal: Si

Original:   25-08-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 24-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

- **geo_VectorMatrizSimInterna.m**:

```
Funcion:    geo_VectorMatrizSimInterna.m

Proposito:  Crea una matriz simetrica a partir de vectores fila con la
            descripcion de sus elementos. Esta funcion no tiene ningun tipo de
            control interno de errores, a fin de hacer mas rapida su ejecicion

Entradas:   Grupos de dos argumentos, de la siguiente forma:
            - Vector fila o matriz con la definicion de la parte triangular
              superior de una matriz simetrica (cada una de sus filas en el
              caso de haber pasado una matriz)
            - Numero indicador de la dimensiones de la matriz correspondiente
              a la definicion anterior, un numero natural. Solo hace falta un
              valor porque la matriz es cuadrada. Si el numero pasado coincide
              con el numero de elementos del vector o las columnas de la
              matriz, se considera que la matriz es diagonal

Salidas:    - Matriz diagonal por bloques, cada uno de los cuales es la matriz
              que define cada elemento de definicion pasado
              De
                          matriz1=[1 2]; matriz2=[1 2 3;4 5 6]; y matriz3=7;
              al ejecutar
                          geo_VectorMatrizSim(matriz1,2,matriz2,2,matriz3,1);
```

```
              se obtiene
                          matriz=[1  0  0  0  0  0  0
                                  0  2  0  0  0  0  0
                                  0  0  1  2  0  0  0
                                  0  0  2  3  0  0  0
                                  0  0  0  0  4  5  0
                                  0  0  0  0  5  6  0
                                  0  0  0  0  0  0  7];

Ejemplo:    matriz=geo_VectorMatrizSim(matriz1,3,matriz2,5,matriz3,1);

Usa geoVarGlobal: No

Original:   25-08-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 24-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

## 10.6. Carpeta 06-estadistica

En esta carpeta se almacenan las funciones relacionadas con el trabajo con funciones estadísticas.

- **geo_ChiCuadInv.m**:

```
Funcion:    geo_ChiCuadInv.m

Proposito:  Calcula el inverso de la CDF (cumulative distribution function)
            para la distribucion Chi-Cuadrado de Pearson

Entradas:   - Probabilidad, numero real en el dominio [0 1]. Escalar o vector
              columna
            - Grados de libertad, numero natural

Salidas:    - Inverso de la CDF, escalar o vector columna

Ejemplo:    x=geo_ChiCuadInv(prob,gradosLibertad);

Usa geoVarGlobal: Si

Nota:       Esta funcion ha sido programada para proporcionar compatibilidad
            entre GNU Octave y MatLab

Original:   23-03-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 25-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_CoeficienteCorrelacion.m**:

```
Funcion:    geo_CoeficienteCorrelacion.m

Proposito:  Calcula el coeficiente de correlacion entre dos variables
```

```
Entradas:   Tres posibilidades:
              - Un argumento de entrada:
                - Matriz varianza-covarianza de un grupo de parametros
              - Dos argumentos de entrada:
                - Vector fila o matriz con la definicion de la matriz
                  varianza-covarianza (cada una de sus filas en el caso de
                  haber pasado una matriz):
                  varCov=[Sa^2 Sab Sac ... Sb^2 Sbc ... Sn^2];
                - Numero indicador de la dimensiones de la matriz
                  correspondiente a la definicion anterior, un numero natural
                  Solo hace falta un valor porque la matriz
                  varianza-covarianza es cuadrada. Si el numero pasado
                  coincide con el numero de elementos del vector o las
                  columnas de la matriz, se considera que la matriz es
                  diagonal
              - Tres argumentos de entrada:
                - Escalar o vector columna que contiene el valor de la
                  covarianza entre las variables
                - Escalar o vector columna que contiene la varianza de la
                  primera variable
                - Escalar o vector columna que contiene la varianza de la
                  segunda variable

Salidas:    - Para un argumento de entrada:
              - Matriz de la mismas dimensiones que la matriz de entrada que
                contiene en su parte triangular superior los coeficientes de
                correlacion entre cada par de variables. Los elementos de la
                diagonal principal se rellenan con el valor 1
            - Para dos argumentos de entrada:
              - Vector fila o matriz con la definicion de una matriz identica
                a la del apartado anterior (cada una de sus filas en el caso
                de haber pasado una matriz). Aunque la matriz de entrada sea
                diagonal y, por lo tanto, todos los coeficientes de
                correlacion igual a 0.0, la matriz de salida sera un vector de
                definicion de matriz completa
            - Para tres argumentos de entrada:
              - Escalar o vector columna que contiene los coeficientes de
                correlacion para cada par de variables

Ejemplo:    coefCorr=geo_CoeficienteCorrelacion(matriz);
            coefCorr=geo_CoeficienteCorrelacion(vector,tamanyo);
            coefCorr=geo_CoeficienteCorrelacion(cov12,var1,var2);

Usa geoVarGlobal: Si

Original:   03-01-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Correccion de bug en el caso del trabajo con dos argumentos de
              entrada. Cuando el primer argumento eran varias matrices, solo
              se calculaban los coeficientes de correlacion para la primera
              Adicion de sentencias de test
```

- **geo_FInv.m**:

```
Funcion:    geo_FInv.m
```

```
Proposito:  Calcula el inverso de la CDF (cumulative distribution function)
            para la distribucion F de Snedecor

Entradas:   - Probabilidad, numero real en el dominio [0 1]. Escalar o vector
              columna
            - Grados de libertad del numerador, numero natural
            - Grados de libertad del denominador, numero natural

Salidas:    - Inverso de la CDF, escalar o vector columna

Ejemplo:    x=geo_FInv(prob,vNumerador,vDenominador);

Usa geoVarGlobal: Si

Nota:       Esta funcion ha sido programada para proporcionar compatibilidad
            entre GNU Octave y MatLab

Original:   23-03-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_NormInv.m**:

```
Funcion:    geo_NormInv.m

Proposito:  Calcula el inverso de la CDF (cumulative distribution function)
            para la distribucion normal

Entradas:   - Probabilidad, numero real en el dominio [0 1]. Escalar o vector
              columna
            Argumentos opcionales:
              - Media de la distribucion normal, escalar o vector columna. Por
                defecto 0
              - Varianza de la distribucion normal, escalar o vector columna,
                numero positivo. Por defecto 1

Salidas:    - Inverso de la CDF, escalar o vector columna

Ejemplo:    x=geo_NormInv(prob);
            x=geo_NormInv(prob,mu,var);

Usa geoVarGlobal: Si

Nota:       Esta funcion ha sido programada para proporcionar compatibilidad
            entre GNU Octave y MatLab

Original:   21-03-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_TInv.m**:

```
Funcion:    geo_TInv.m

Proposito:  Calcula el inverso de la CDF (cumulative distribution function)
            para la distribucion t-Student

Entradas:   - Probabilidad, numero real en el dominio [0 1]. Escalar o vector
              columna
            - Grados de libertad, numero natural

Salidas:    - Inverso de la CDF, escalar o vector columna

Ejemplo:    x=geo_TInv(prob,gradosLibertad);

Usa geoVarGlobal: Si

Nota:       Esta funcion ha sido programada para proporcionar compatibilidad
            entre GNU Octave y MatLab

Original:   23-03-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_TauInv.m**:

```
Funcion:    geo_TauInv.m

Proposito:  Calcula el inverso de la CDF (cumulative distribution function)
            para la distribucion Tau, tal como se define en [CA-CNDA-1987],
            pag. 76

Entradas:   - Probabilidad, numero real en el dominio [0 1]. Escalar o vector
              columna
            - Grados de libertad, numero natural mayor que 1

Salidas:    - Inverso de la CDF, escalar o vector columna

Ejemplo:    x=geo_TauInv(prob,gradosLibertad);

Usa geoVarGlobal: Si

Original:   23-03-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_TestGlobModChiCuad.m**:

```
Funcion:    geo_TestGlobModChiCuad.m
```

```
Proposito:  Realiza el test global de un modelo de ajuste minimos cuadrados
            utilizando la distribucion Chi-Cuadrado de Pearson

Entradas:   - Varianza del observable de peso unidad a priori, escalar
            - Varianza del observable de peso unidad a posteriori, escalar
            - Grados de libertad, numero natural
            - Nivel de probabilidad, escalar o vector columna
            Argumento opcional:
              - Numero identificador del numero de colas de la distribucion
                que se utilizaran en el test. Tres posibilidades:
                - 0: Se realiza un test de una cola y otro de dos colas, este
                     es el valor por defecto
                - 1: Se realiza un test de una cola
                - 2: Se realiza un test de dos colas

Salidas:    - Matriz con tantas filas como filas tenga el vector de niveles de
              probabilidad pasado. En cada columna se almacena lo siguiente:
              - Col. 1: Valor del nivel de probabilidad pasado
              - Col. 2: Valor del estadistico gLib*(varPos/varPri)
              Si el quinto argumento de entrada vale 1:
                - Col. 3: Valor de la distribucion Chi para los niveles de
                          probabilidad y los grados de libertad pasados
                - Col. 4: Indicador de si se ha pasado el test: 0/1 -> no/si
              Si el quinto argumento de entrada vale 2:
                - Col. 3: Valor de la distribucion Chi por la izquierda para
                          los niveles de probabilidad y los grados de libertad
                          pasados
                - Col. 4: Valor de la distribucion Chi por la derecha para
                          los niveles de probabilidad y los grados de libertad
                          pasados
                - Col. 5: Indicador de si se ha pasado el test: 0/1 -> no/si
              Si el quinto argumento de entrada vale 0:
                - Col. 3: Valor de la distribucion Chi para los niveles de
                          probabilidad y los grados de libertad pasados para
                          el test de una cola
                - Col. 4: Indicador de si se ha pasado el test de una cola:
                          0/1 -> no/si
                - Col. 5: Valor de la distribucion Chi por la izquierda para
                          los niveles de probabilidad y los grados de libertad
                          pasados, para el test de dos colas
                - Col. 6: Valor de la distribucion Chi por la derecha para
                          los niveles de probabilidad y los grados de libertad
                          pasados, para el test de dos colas
                - Col. 7: Indicador de si se ha pasado el test de dos colas:
                          0/1 -> no/si

Ejemplo:    resultado=geo_TestGlobModChiCuad(varPri,varPos,gLib,alfa);
            resultado=geo_TestGlobModChiCuad(varPri,varPos,gLib,alfa,colas);

Usa geoVarGlobal: Si

Original:   26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

## 10.7. Carpeta 07-elipsesError

En esta carpeta se gurdan las funciones relacionadas con el cálculo de elipses de error absolutas.

- **geo_ElipseErrorPlano.m**:

```
Funcion:    geo_ElipseErrorPlano.m

Proposito:  Calcula elipses de error en dos dimensiones a partir de matrices
            varianza-covarianza

Entradas:   - Vector o matriz que contiene informacion sobre las matrices
              varianza-covarianza. Dos posibilidades:
              - Vector (o matriz) de dos columnas. Cada fila contiene la
                expresion de una matriz varianza-covarianza diagonal:
                varCov=[Sx^2 Sy^2];
              - Vector (o matriz) de tres columnas. Cada fila contiene la
                expresion de una matriz varianza-covarianza completa:
                varCov=[Sx^2 Sxy Sy^2];

Salidas:    - Vector o matriz, de tamano nx3, que contiene los datos acerca de
              las elipses de error calculadas. Su estructura es la siguiente:
              elipses=[a1  b1  ang1
                       ............
                       an  bn  angn];
              Cada fila corresponde a una elipse:
              - Col. 1: Semieje mayor de la elipse, en las mismas unidades
                        (raiz cuadrada) que las de la matriz de
                        varianzas-covarianzas
              - Col. 2: Semieje menor de la elipse, en las mismas unidades
                        (raiz cuadrada) que las de la matriz de
                        varianzas-covarianzas
              - Col. 3: Angulo en sentido antihorario (en el dominio ]-pi pi],
                        medido a partir de la parte positiva del eje X) del
                        semieje mayor de la elipse, en radianes

Ejemplo:    [elipses]=geo_ElipseErrorPlano(varCov);

Usa geoVarGlobal: No

Nota:       Hay que tener en cuenta un detalle en el caso de que queramos
            calcular las elipses a partir de matrices varianza-covarianza de
            coordenadas geodesicas. Si las matrices de este tipo de
            coordenadas vienen dadas en la forma
            varCov=[Slat^2 Slatlon Slon^2], el angulo calculado tendra su
            origen en los meridianos, lo que puede conllevar a errores al
            calcular el acimut. Si hacemos el cambio
            varCov=[Slon^2 Slatlon Slat^2], el angulo tendra su origen en los
            paralelos y el acimut sera identico al calculado si se pasara
            varCov=[Sx^2 Sxy Sy^2]

Original:   12-08-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 27-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

## 10.8. Carpeta 08-datosBinarios

### 10.8.1. Carpeta 01-tipoDatoBinario

En esta carpeta se almacenan las funciones de verificación de los tipos de datos binarios de GEO.

- **geo_EsTipoAlmGeo.m**:

```
Funcion:    geo_EsTipoAlmGeo.m

Proposito:  Comprueba si el argumento pasado es un identificador de tipo de
            almacenamiento de dato binario utilizado por GEO (alguno de los
            almacenados en geoVarGlobal.tipoAlmacenBinarioGeo{})

Entradas:   - Una variable, como cadena de texto

Salidas:    - Un escalar, cuyos valores pueden ser:
              - 1, si el argumento correspondiente es un identificador de tipo
                de almacenamiento de dato binario de GEO
              - 0, en caso contrario

Ejemplo:    esTipoAlmGeo=geo_EsTipoAlmGeo(variable);

Usa geoVarGlobal: Si

Original:   24-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 01-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Corregido bug que hacia que si el argumento de entrada no era de
              tipo texto la funcion no devolviese nada
              Adicion de sentencias de test
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Corregido bug que hacia que la funcion diera error si el nombre
              del tipo de dato pasado contenia espacios al principio o al
              final
```

- **geo_EsTipoBinarioGeo.m**:

```
Funcion:    geo_EsTipoBinarioGeo.m

Proposito:  Comprueba si el argumento pasado es un identificador de tipo de
            dato binario utilizado por GEO (alguno de los almacenados en
            geoVarGlobal.nombreDatoBinGeo{})

Entradas:   - Una variable, como cadena de texto

Salidas:    - Un escalar, cuyos valores pueden ser:
              - 1, si el argumento correspondiente es un identificador de tipo
                de dato binario de GEO
              - 0, en caso contrario

Ejemplo:    esTipoBinarioGeo=geo_EsTipoBinarioGeo(variable);

Usa geoVarGlobal: Si
```

```
Original:   24-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 01-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Corregido bug que hacia que si el argumento de entrada no era de
              tipo texto la funcion no devolviese nada
              Adicion de sentencias de test
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Corregido bug que hacia que la funcion diera error si el nombre
              del tipo de dato pasado contenia espacios al principio o al
              final
```

- **geo_LonTipoBinarioGeo.m**:

```
Funcion:    geo_LonTipoBinarioGeo.m

Proposito:  Calcula el tamanyo (en bytes) de cualquier tipo de dato binario
            utilizado por GEO

Entradas:   - El nombre de un tipo de dato binario utilizado por GEO, como
              cadena de texto (alguno de los almacenados en
              geoVarGlobal.nombreDatoBinGeo{})

Salidas:    - El tamanyo del tipo de dato en bytes, como numero natural

Ejemplo:    tamanyo=geo_LonTipoBinarioGeo('ENTERO*2');

Usa geoVarGlobal: Si

Original:   24-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 01-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_TipoAlmGeoTipoAlmOctave.m**:

```
Funcion:    geo_TipoAlmGeoTipoAlmOctave.m

Proposito:  Asigna el tipo de almacenamiento de datos binarios de
            GNU/Octave-MatLab que corresponde con un tipo de dato binario de
            GEO, es decir, relaciona geoVarGlobal.tipoAlmacenBinarioGeo{} con
            geoVarGlobal.tipoAlmacenBinarioOctave{}

Entradas:   - El nombre de un tipo de almacenamiento de datos binarios
              utilizado por GEO, como cadena de texto

Salidas:    - Nombre del tipo de almacenamiento de datos correspondiente en
              GNU/Octave-MatLab, como cadena de texto

Ejemplo:    tipoOctave=geo_TipoAlmGeoTipoAlmOctave('BIG-ENDIAN');
```

```
Usa geoVarGlobal: Si

Original:   24-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 01-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Corregido bug que hacia que la funcion diera error si el nombre
              del tipo de dato pasado contenia espacios al principio o al
              final
```

- **geo_TipoBinGeoTipoBinC.m**:

```
Funcion:    geo_TipoBinGeoTipoBinC.m

Proposito:  Asigna el tipo de dato binario del lenguaje C/C++ que corresponde
            con un tipo de dato binario de GEO y da el tamanyo del tipo de GEO
            en bytes

Entradas:   - Nombre de un tipo de dato binario utilizado por GEO, como cadena
              de texto

Salidas:    - Nombre del tipo de dato correspondiente al lenguaje C/C++, como
              cadena de texto
            - Tamanyo del tipo de GEO en bytes, como numero natural

Ejemplo:    [tipoC,tamanyo]=geo_TipoBinGeoTipoBinC('ENTERO*2');

Usa geoVarGlobal: Si

Original:   24-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 01-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Corregido bug que hacia que la funcion diera error si el nombre
              del tipo de dato pasado contenia espacios al principio o al
              final
```

- **geo_TipoBinGeoTipoBinOctave.m**:

```
Funcion:    geo_TipoBinGeoTipoBinOctave.m

Proposito:  Asigna el tipo de dato binario de GNU/Octave-MatLab que
            corresponde con un tipo de dato binario de GEO y da el tamanyo del
            tipo de GEO en bytes

Entradas:   - Nombre de un tipo de dato binario utilizado por GEO, como cadena
              de texto

Salidas:    - Nombre del tipo de dato correspondiente en GNU/Octave-MatLab,
```

```
              como cadena de texto
            - Tamanyo del tipo de GEO en bytes, como numero natural

Ejemplo:    [tipoOctave,tamanyo]=geo_TipoBinGeoTipoBinOctave('ENTERO*2');

Usa geoVarGlobal: Si

Original:   24-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 01-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Corregido bug que hacia que la funcion diera error si el nombre
              del tipo de dato pasado contenia espacios al principio o al
              final
```

### 10.8.2. Carpeta 02-conversorBinario

En esta carpeta se almacenan funciones para la conversión de ficheros de formato ASCII a formato binario de GEO.

- **geo_AsciiBin.m**:

```
Funcion:    geo_AsciiBin.m

Proposito:  Convierte un fichero que contiene datos numericos de formato ASCII
            a formato binario. Los datos han de estar separados por espacios
            en el fichero original

Entradas:   - Nombre del fichero de texto original, como cadena de texto
            - Nombre del fichero binario de salida, como cadena de texto
            - Nombre del dato binario de GEO, como cadena de texto. Los tipos
              de dato binario permitidos estan definidos en la estructura
              global geoVarGlobal.nombreDatoBinGeo{}
            - Argumentos opcionales:
              - Numero de lineas de la cabecera del fichero original que seran
                omitidas, como numero entero >= 0. En el caso de tener
                activado en GNU Octave el uso de funciones escritas en C/C++,
                las lineas de cabecera no pueden tener mas de 250 caracteres
              - Escalar que almacena un factor de multiplicacion para aplicar
                a los datos del fichero original. Por defecto, este argumento
                vale 1.0
              - Codigo del formato de escritura de los datos binarios en el
                fichero destino, como cadena de texto. Los formatos de
                escritura de dato binario permitidos estan definidos en la
                estructura global geoVarGlobal.tipoAlmacenBinarioGeo{}
                Si se omite este argumento, la funcion considera el tipo de
                almacenamiento de la maquina en la que esta corriendo el
                programa

Salidas:    - Fichero binario con los valores del fichero de texto. Este
              fichero se creara sin ningun tipo de cabecera, comenzara
              directamente con los datos y estos estaran almacenados uno a
              continuacion de otro, sin bytes en blanco entre ellos y
```

```
              manteniendo el orden de almacenamiento del fichero original. Se
              almacena en el directorio de trabajo.
            - Numero de elementos escritos en el fichero binario

Ejemplo:    [nPuntos]=geo_AsciiBin(fOrig,fSal,nombreDato);
            [nPuntos]=geo_AsciiBin(fOrig,fSal,cabecera,nombreDato,fEscala,
            endian);

Usa geoVarGlobal: Si

Original:   24-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 02-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Corregido bug que hacia que pudiese aceptarse un numero real
              positivo como cuarto argumento de entrada
              Adicion de sentencias de test
```

- **geo_BinAscii.m**:

```
Funcion:    geo_BinAscii.m

Proposito:  Convierte un fichero que contiene datos numericos en formato
            binario a formato ASCII

Entradas:   - Nombre del fichero binario original, como cadena de texto. Este
              fichero no tendra ningun tipo de cabecera, comenzara
              directamente con los datos y estos estaran almacenados uno a
              continuacion de otro, sin bytes en blanco entre ellos
            - Nombre del fichero de texto de salida, como cadena de texto
            - Nombre del dato binario de GEO correspondiente a los datos
              almacenados en el fichero binario, como cadena de texto. Los
              tipos de dato binario permitidos estan definidos en la
              estructura global geoVarGlobal.nombreDatoBinGeo{}
            - Formato de escritura de los datos en cada linea del fichero
              ASCII, como cadena de texto. Por ejemplo:
              '%d' -> numero entero
              '%10d' -> numero entero ocupando un espacio de 10 caracteres
              '%.3f' -> numero real con tres lugares decimales
              '%10.4f' -> numero real ocupando un espacio de 10 caracteres, de
                          los cuales los 4 ultimos son decimales
              El caracter de fin de linea y retorno de carro ('\n') no se
              incluira en la cadena de formato pasada porque sera anyadida en
              la ejecucion de la funcion
            - Argumentos opcionales:
              - Escalar que almacena un factor de multiplicacion para aplicar
                a los datos del fichero original. Por defecto, este argumento
                vale 1.0
              - Codigo del formato de escritura de los datos binarios en el
                fichero destino, como cadena de texto. Los formatos de
                escritura de dato binario permitidos estan definidos en la
                estructura global geoVarGlobal.tipoAlmacenBinarioGeo{}
                Si se omite este argumento, la funcion considera el tipo de
                almacenamiento de la maquina en la que esta corriendo el
                programa
```

```
Salidas:    - Fichero ASCII con los valores en el formato pasado. Los datos
              siguen manteniendo el orden de almacenamiento del fichero
              original. Se almacena en el directorio de trabajo
            - Numero de puntos convertidos

Ejemplo:    [nPuntos]=geo_BinAscii(fOrig,fSal,nombreDato,formato);
            [nPuntos]=geo_BinAscii(fOrig,fSal,nombreDato,formato,fEscala,
            endian);

Usa geoVarGlobal: Si

Original:   24-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 02-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Corregido bug que hacia que pudiese aceptarse un numero real
              positivo como cuarto argumento de entrada
              Adicion de sentencias de test
```

## 10.9. Carpeta 09-interpMalla

En esta carpeta se almacenan las funciones relacionadas con la interpolación de datos en mallas cuadradas.

- **geo_EsEstructuraMalla.m**:

```
Funcion:    geo_EsEstructuraMalla.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba si el argumento pasado es una estructura malla

Entradas:   - Una variable

Salidas:    - 1, si el argumento de entrada es una estructura malla
            - Un error, en caso contrario

Ejemplo:    esMalla=geo_EsEstructuraMalla(malla);

Funciones utilizadas de GEO: - geo_IndGeoVarGlobal.m
                             - geo_EsTexto.m
                             - geo_EsEntero.m
                             - geo_EsEscalar.m
                             - geo_ControlDominio.m

Funciones relacionadas de GEO: - geo_EstructuraMalla.m

Ultima revision: 25-02-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_EstructuraMalla.m**:

```
Funcion:    geo_EstructuraMalla.m
```

```
Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Crea una estructura 'malla', para su utilizacion en la funcion
            'geo_ExtraeVentanaMalla.m'

Entradas:   - Codigo identificador del sistema de coordenadas en que vienen
              expresados los puntos dato:
              0/1->coordenadas cartesianas/coordenadas geodesicas
            - Limite norte de la malla de datos: si el sistema de coordenadas
              es geodesico, latitud en radianes; si es cartesiano, maximo
              valor de la coordenada Y
            - Limite sur de la malla de datos: si el sistema de coordenadas es
              geodesico, latitud en radianes; si es cartesiano, minimo valor
              de la coordenada Y
            - Limite este de la malla de datos: si el sistema de coordenadas
              es geodesico, longitud en radianes; si es cartesiano, maximo
              valor de la coordenada X
            - Limite oeste de la malla de datos: si el sistema de coordenadas
              es geodesico, longitud en radianes; si es cartesiano, minimo
              valor de la coordenada X
            - Espaciado (paso de malla) entre los puntos dato (en valor
              absoluto) en la direccion norte-sur (si el sistema de
              coordenadas es geodesico, en radianes)
            - Espaciado (paso de malla) entre los puntos dato (en valor
              absoluto) en la direccion este-oeste (si el sistema de
              coordenadas es geodesico, en radianes)
            - Indicador de cobertura global en latitud: 0/1->no/si
            - Indicador de cobertura global en longitud: 0/1->no/si

Salidas:    - Una estructura malla, cuyos campos son:
              - malla.coorPuntos: indicador del sistema de coordenadas en que
                                  vienen expresados los puntos dato
              - malla.cobGlobLat: numero identificador de cobertura global en
                                  la direccion norte-sur
              - malla.cobGlobLon: numero identificador de cobertura global en
                                  la direccion este-oeste
              - malla.limNorte:   limite norte de la malla de puntos
              - malla.limSur:     limite sur de la malla de puntos
              - malla.limEste:    limite este de la malla de puntos
              - malla.limOeste:   limite oeste de la malla de puntos
              - malla.pasoLat:    paso de malla en la direccion norte-sur
              - malla.pasoLon:    paso de malla en la direccion este oeste
              Las unidades de todos los campos de la estructura son las mismas
              que las unidades de entrada de cada parametro

Ejemplo:    malla=geo_EstructuraMalla(sistCoor,limNorte,limSur,limEste,
            limOeste,pasoLat,pasoLon,cobGlobLat,cobGlobLon);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_EsEstructuraMalla.m

Nota: En esta funcion no se controlan ni el tipo ni el dominio de los datos de
      entrada

Ultima revision: 25-02-07
```

```
Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ExtraeDatosFicheroMalla.m**:

```
Funcion:    geo_ExtraeDatosFicheroMalla.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Extrae datos de un fichero binario en formato GEO

Entradas:   - Tamano de la ventana de interpolacion, 1 o numero par
            - Indicador de cobertura global en latitud: 0/1->no/si
            - Indicador de cobertura global en longitud: 0/1->no/si
            - Numero de nodos de la malla en latitud, numero entero
            - Numero de nodos de la malla en longitud, numero entero
            - Escalar o vector columna que almacena la coordenada X del nodo
              anterior en longitud al punto de trabajo con respecto a la
              esquina noroeste del cuadrado que define la malla de puntos dato
              La coordenada X se mide en la direccion del paralelo
            - Escalar o vector columna que almacena la coordenada Y del nodo
              anterior en latitud al punto de trabajo con respecto a la
              esquina noroeste del cuadrado que define la malla de puntos dato
              La coordenada Y se mide en la direccion del meridiano
            - Nombre del tipo de dato guardado en el fichero binario, cadena
              de texto
            - Longitud del tipo de dato guardado en el fichero binario, numero
              natural
            - Codigo identificador en GNU/Octave-MatLab del tipo de
              almacenamiento de datos binarios
            - Ruta completa del fichero binario que contiene los datos

Salidas:    - Matriz de tamano (VENTANAxN)x(VENTANA) que contiene el valor del
              dato del fichero para cada ventana de interpolacion

Ejemplo:    matVent = geo_ExtraeDatosFicheroMalla(ventana,cobGlobLat,
            cobGlobLon,puntosLat,puntosLon,preLat,preLon,nombreDato,
            longitudDato,tipoAlmac,nombreFichero);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_ExtraeVentanaMalla.m
                               - geo_InterpolaMalla.m

Nota : En esta funcion no se controla ni el tipo de datos de entrada ni si
       estos estan en el dominio correcto

Ultima revision: 03-03-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ExtraeVentanaMalla.m**:

```
Funcion:    geo_ExtraeVentanaMalla.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
Proposito:  Extrae ventanas de interpolacion de un fichero binario en formato
            de GEO

Entradas:   - Latitud geodesica (en radianes) o coordenada Y del punto de
              trabajo (escalar o vector columna)
            - Longitud geodesica (en radianes) o coordenada X del punto de
              trabajo (escalar o vector columna)
            - Codigo identificador del sistema de coordenadas en que vienen
              expresados los puntos dato:
              0/1->coordenadas cartesianas/coordenadas geodesicas
            - Tamano de la ventana de interpolacion, 1 o numero par entre 2 y
              10
            - Limite norte de la malla de datos: si el sistema de coordenadas
              es geodesico, latitud en radianes; si es cartesiano, maximo
              valor de la coordenada Y
            - Limite sur de la malla de datos: si el sistema de coordenadas es
              geodesico, latitud en radianes; si es cartesiano, minimo valor
              de la coordenada Y
            - Limite este de la malla de datos: si el sistema de coordenadas
              es geodesico, longitud en radianes; si es cartesiano, maximo
              valor de la coordenada X
            - Limite oeste de la malla de datos: si el sistema de coordenadas
              es geodesico, longitud en radianes; si es cartesiano, minimo
              valor de la coordenada X
            - Espaciado (paso de malla) entre los puntos dato (en valor
              absoluto) en la direccion norte-sur (si el sistema de
              coordenadas es geodesico, en radianes)
            - Espaciado (paso de malla) entre los puntos dato (en valor
              absoluto) en la direccion este-oeste (si el sistema de
              coordenadas es geodesico, en radianes)
            - Indicador de cobertura global en latitud: 0/1->no/si
            - Indicador de cobertura global en longitud: 0/1->no/si
            - Nombre del tipo de dato de GEO utilizado para almacenar los
              datos en el fichero binario correspondiente. Los nombres del
              tipo de dato se encuentran definidos en la estructura global
              geoVarGlobal.nombreDatoBinGeo{i}, como cadena de texto
            - Nombre del tipo de almacenamiento de los datos en el fichero
              binario correspondiente. Los nombres identificadores del tipo de
              almacenamiento se encuentran definidos en la estructura global
              geoVarGlobal.tipoAlmacenBinarioGeo{i}, como cadena de texto
            - Ruta completa del fichero binario que contiene los datos, como
              cadena de texto

Salidas:    - Matriz de tamano (VENTANAxN)x(VENTANA) que contiene el valor del
              dato del fichero para cada ventana de interpolacion
            - Escalar o vector columna que almacena la coordenada X del punto
              de trabajo con respecto a la esquina sudoeste de la ventana de
              interpolacion. La coordenada X se mide en la direccion del
              paralelo
            - Escalar o vector columna que almacena la coordenada Y del punto
              de trabajo con respecto a la esquina sudoeste de la ventana de
              interpolacion. La coordenada Y se mide en la direccion del
              meridiano

Ejemplo:    [matVent,posX,posY]=geo_ExtraeVentanaMalla(latGeo,lonGeo,sistCoor,
            ventana,limNorte,limSur,limEste,limOeste,pasoLat,pasoLon,
            cobGlobLat,cobGlobLon,nombreDato,tipoAlmacen,nombreFichero);
```

```
Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_EstructuraMalla.m
                             - geo_EsEstructuraMalla.m
                             - geo_EsTipoBinarioGeo.m
                             - geo_EsTipoAlmGeo.m
                             - geo_EsTexto.m
                             - geo_ControlDominioInterpMalla.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_PonAcimutDominio.m
                             - geo_TipoBinGeoTipoBinC.m
                             - geo_QuitaEspacios.m
                             - geo_TipoBinGeoTipoBinOctave.m
                             - geo_TipoAlmGeoTipoAlmOctave.m
                             - geo_ExtraeDatosFicheroMalla.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_ExtraeDatosFicheroMalla.m
                               - geo_InterpolaMalla.m

Ultima revision: 04-03-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_InterpolaMalla.m**:

```
Funcion:    geo_InterpolaMalla.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Realiza una interpolacion por el metodo del vecino mas proximo,
            bilineal o spline cubica natural en una matriz de datos

Entradas:   - Matriz de tamano (VENTANAxN)x(VENTANA) que contiene el valor del
              dato del fichero para cada ventana de interpolacion
            - Escalar o vector columna que almacena la coordenada X del punto
              de trabajo con respecto a la esquina sudoeste de la ventana de
              interpolacion. La coordenada X se mide en la direccion del
              paralelo
            - Escalar o vector columna que almacena la coordenada Y del punto
              de trabajo con respecto a la esquina sudoeste de la ventana de
              interpolacion. La coordenada Y se mide en la direccion del
              meridiano
            - Tamano de la ventana de interpolacion, 1 o numero par. Si el
              numero es 1 se utiliza el vecino mas proximo como solucion, si
              es 2 se realiza una interpolacion bilineal. En otro caso, la
              interpolacion se realiza por medio de splines cubicos

Salidas:    - Valor interpolado (escalar o vector columna), en las mismas
              unidades que las de los valores dato de la matriz de entrada

Ejemplo:    valor = geo_InterpolaMalla(matrizVentana,posX,posY,ventana);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_ExtraeDatosFicheroMalla.m
                               - geo_ExtraeVentanaMalla.m
```

```
Nota 1: En esta funcion no se controla ni el tipo de datos de entrada ni si
        estos estan en el dominio correcto
Nota 2: Si la interpolacion se realiza por el metodo del vecino mas proximo y
        el punto de trabajo esta a la misma distancia a dos o mas puntos del
        cuadrilatero que lo encierra, el valor obtenido sera el del punto del
        cuadrilatero al que se haya calculado antes la distancia, siendo el
        orden de calculo el sentido de las agujas del reloj, comenzando por la
        esquina NW

Ultima revision: 04-03-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 10.10. Carpeta 10-interpNubePuntos

En esta carpeta se almacenan las funciones relacionadas con la interpolación de datos en nubes de puntos.

- **geo_InterpolaInvDistEspacio.m**:

```
Funcion:    geo_InterpolaInvDistEspacio.m

Proposito:  Interpola atributos en una nube de puntos en el espacio,
            ponderando con una potencia del inverso de la distancia y calcula
            una estimacion del error cometido (varianza)

Entradas:   - Vector (o matriz) de tres columnas que contiene las coordenadas
              en el espacio de los puntos a interpolar. Cada fila corresponde
              a un punto:
              - Col. 1: Coordenada X del punto a interpolar
              - Col. 2: Coordenada Y del punto a interpolar
              - Col. 3: Coordenada Z del punto a interpolar
            - Matriz de Nx4 elementos que almacena las coordenadas y atributo
              de los puntos de la nube. Cada fila corresponde a un punto:
              - Col. 1: Coordenada X
              - Col. 2: Coordenada Y
              - Col. 3: Coordenada Z
              - Col. 4: Atributo del punto
            - Numero natural, que indica el numero de puntos de la nube (los
              mas cercanos) que han de ser tenidos en cuenta para realizar la
              interpolacion
            - Numero real, que indica la potencia a la que se elevara el
              inverso de la distancia para ponderar la media. Si este
              argumento vale 0.0, se utilizara la media aritmetica para la
              interpolacion
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza de las coordenadas del punto de
                interpolacion. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada. Queda asi expresada
                  una matriz varianza-covarianza diagonal:
                  varCov=[Sxp^2 Syp^2 Szp^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y su correlacion
                  Queda asi expresada (por filas) una matriz
```

```
                    varianza-covarianza completa:
                    varCov=[Sxp^2 Sxpyp Sxpzp Syp^2 Sypzp Szp^2];
                - Vector o matriz que contiene informacion sobre la matriz
                  varianza-covarianza de las coordenadas de cada punto de la
                  nube. Dos posibilidades:
                  - Vector (o matriz) de cuatro columnas, cada una de las cuales
                    contiene la varianza de cada coordenada y del atributo
                    Queda asi expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sx^2 Sy^2 Sz^2 Sa^2];
                  - Vector (o matriz) de diez columnas, cada una de las cuales
                    contiene la varianza de cada coordenada y del atributo y su
                    correlacion. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sx^2 Sxy Sxz Sxa Sy^2 Syz Sya Sz^2 Sza Sa^2];
                Si, en vez de las matrices varianza-covarianza, se envian dos
                variables de valor -1, no se hace el calculo del error

Salidas:    - Atributo interpolado en las coordenadas pasadas (escalar o
              vector columna), en las mismas unidades que los atributos de los
              puntos de la nube
            - Varianza del atributo interpolado (escalar o vector columna), en
              las mismas unidades (al cuadrado) que las del atributo
              En el caso de no haber pasado datos para el calculo del error
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [A,varA]=geo_InterpolaInvDistEspacio(interp,nube,nPtos,pot);
            [A,varA]=geo_InterpolaInvDistEspacio(interp,nube,nPtos,pot,-1,-1);
            [A,varA]=geo_InterpolaInvDistEspacio(interp,nube,nPtos,pot,
            varInterp,varNube);

Usa geoVarGlobal: Si

Original:   18-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 22-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_InterpolaInvDistPlano.m**:

```
Funcion:    geo_InterpolaInvDistPlano.m

Proposito:  Interpola atributos en una nube de puntos en el plano, ponderando
            con una potencia del inverso de la distancia y calcula una
            estimacion del error cometido (varianza)

Entradas:   - Vector (o matriz) de dos columnas que contiene las coordenadas
              en el plano de los puntos a interpolar. Cada fila corresponde a
              un punto:
              - Col. 1: Coordenada X del punto a interpolar
              - Col. 2: Coordenada Y del punto a interpolar
            - Matriz de Nx3 elementos que almacena las coordenadas y atributo
              de los puntos de la nube. Cada fila corresponde a un punto:
              - Col. 1: Coordenada X
              - Col. 2: Coordenada Y
              - Col. 3: Atributo del punto
```

```
              - Numero natural, que indica el numero de puntos de la nube (los
                mas cercanos) que han de ser tenidos en cuenta para realizar la
                interpolacion
              - Numero real que indica la potencia a la que se elevara el
                inverso de la distancia para ponderar la media. Si este
                argumento vale 0.0, se utilizara la media aritmetica para la
                interpolacion
              Argumentos opcionales:
                - Vector o matriz que contiene informacion sobre la matriz
                  varianza-covarianza de las coordenadas del punto de
                  interpolacion. Dos posibilidades:
                  - Vector (o matriz) de dos columnas, cada una de las cuales
                    contiene la varianza de cada coordenada. Queda asi expresada
                    una matriz varianza-covarianza diagonal:
                    varCov=[Sxp^2 Syp^2];
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada coordenada y su correlacion
                    Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sxp^2 Sxpyp Syp^2];
                - Vector o matriz que contiene informacion sobre la matriz
                  varianza-covarianza de las coordenadas de cada punto de la
                  nube. Dos posibilidades:
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada coordenada y del atributo
                    Queda asi expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sx^2 Sy^2 Sz^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada coordenada y del atributo y su
                    correlacion. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
                  Si, en vez de las matrices varianza-covarianza, se envian cuatro
                  variables de valor -1, no se hace el calculo del error

Salidas:      - Atributo interpolado en las coordenadas pasadas (escalar o
                vector columna), en las mismas unidades que los atributos de los
                puntos de la nube
              - Varianza del atributo interpolado (escalar o vector columna), en
                las mismas unidades (al cuadrado) que las del atributo
                En el caso de no haber pasado datos para el calculo del error
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:      [A,varA]=geo_InterpolaInvDistPlano(interp,nube,nPtos,pot);
              [A,varA]=geo_InterpolaInvDistPlano(interp,nube,nPtos,pot,-1,-1);
              [A,varA]=geo_InterpolaInvDistPlano(interp,nube,nPtos,pot,
              varInterp,varNube);

Usa geoVarGlobal: Si

Original:     18-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 22-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_InterpolaTin.m**:

```
Funcion:    geo_InterpolaTin.m

Proposito:  Interpola atributos en una triangulacion de Delaunay y calcula una
            estimacion del error cometido (varianza)

Entradas:   - Vector (o matriz) de dos columnas que contiene las coordenadas
              en el plano de los puntos a interpolar:
              puntoInterp = [xp yp];
              Cada fila corresponde a un punto:
              - Col. 1: Coordenada X del punto a interpolar
              - Col. 2: Coordenada Y del punto a interpolar
            - Matriz de Nx3 elementos que almacena las coordenadas y atributo
              de los puntos de la nube triangulada. Cada fila corresponde a un
              punto:
              - Col. 1: Coordenada X
              - Col. 2: Coordenada Y
              - Col. 3: Atributo del punto
            - Matriz de Tx3 elementos que contiene la triangulacion de
              Delaunay de la nube de puntos pasada. Cada fila describe cada
              uno de los T triangulos que la forman. Cada columna es la
              posicion en la matriz de la nube de puntos que corresponde a
              cada vertice del triangulo
            - Numero natural que indica el numero de puntos de la nube (los
              mas cercanos) que han de ser tenidos en cuenta para realizar la
              interpolacion en el caso de que algun punto a interpolar quede
              fuera de la red de triangulos. En este caso se realizara una
              interpolacion ponderada con una potencia del inverso de la
              distancia a estos puntos mas cercanos
            - Numero real que indica la potencia a la que se elevara el
              inverso de la distancia para ponderar la media
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza de las coordenadas del punto de
                interpolacion. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada. Queda asi expresada
                  una matriz varianza-covarianza diagonal:
                  varCov=[Sxp^2 Syp^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y su correlacion
                  Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxp^2 Sxpyp Syp^2];
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza de las coordenadas y atributo del los
                puntos que forman la nube triangulada. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y del atributo
                  Queda asi expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sx1^2 Sy1^2 Sz1^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y del atributo y su
                  correlacion. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sx1^2 Sx1y1 Sx1z1 Sy1^2 Sy1z1 Sz1^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
```

```
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envian cuatro
                variables de valor -1, no se hace el calculo del error

Salidas:    - Atributo interpolado en las coordenadas pasadas (escalar o
              vector columna), en las mismas unidades que los atributos de los
              puntos de la triangulacion
            - Varianza del atributo interpolado (escalar o vector columna), en
              las mismas unidades (al cuadrado) que las del atributo
              En el caso de no haber pasado datos para el calculo del error
              este argumento de salida tiene el valor -1 (si se ha solicitado)
            - Vector columna indicador de los puntos interpolados que estan
              dentro o fuera de la triangulacion. Tiene el mismo numero de
              filas que la matriz de puntos a interpolar y los valores
              posibles son:
              - 1: Si el punto correspondiente (con el mismo indice) en la
                   matriz esta contenido en la triangulacion. En este caso la
                   interpolacion se reliza en el triangulo que contiene al
                   punto
              - 0: Si el punto correspondiente (con el mismo indice) en la
                   matriz no esta contenido en la triangulacion. En este caso
                   el atributo (y su varianza, si ha lugar) sera interpolado
                   ponderando con una potencia del inverso de la distancia
                   teniendo en cuenta el numero de puntos mas cercanos de la
                   nube de puntos de la triangulacion indicados en el cuarto
                   argumento de entrada de la funcion

Ejemplo:    [Z,varZ,dentro]=geo_InterpolaTin(interp,nube,tin,nPuntos,pot);
            [Z,varZ,dentro]=geo_InterpolaTin(interp,nube,tin,nPuntos,pot,-1,
            -1);
            [Z,varZ,dentro]=geo_InterpolaTin(interp,nube,tin,nPuntos,pot,
            varCovInterp,varCovNube);

Funciones utilizadas de GEO: - geo_EsNatural.m
                             - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_InterpolaTriangulo.m
                             - geo_InterpolaInvDistPlano.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_AltitudMdeTin.m
                               - geo_InterpolaInvDistEspacio.m
                               - geo_InterpolaInvDistPlano.m
                               - geo_InterpolaTriangulo.m

Ultima revision: 18-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_InterpolaTriangulo.m**:

```
Funcion:    geo_InterpolaTriangulo.m

Proposito:  Interpola atributos en un triangulo definido por las coordenadas
            en el plano de cada vertice y calcula una estimacion del error
```

```
            cometido (varianza)

Entradas:   - Vector (o matriz) de dos columnas que contiene las coordenadas
              en el plano de los puntos a interpolar:
              puntoInterp = [xp yp];
              Cada fila corresponde a un punto:
              - Col. 1: Coordenada X del punto a interpolar
              - Col. 2: Coordenada Y del punto a interpolar
            - Vector (o matriz) de tres columnas que contiene las coordenadas
              en el plano y el atributo del primer punto del triangulo:
              puntoTri1 = [x1 y1 z1];
              Cada fila corresponde a un punto:
              - Col. 1: Coordenada X del punto del triangulo
              - Col. 2: Coordenada Y del punto del triangulo
              - Col. 3: Atributo del punto del triangulo (en el ejemplo es la
                        cota)
            - Vector (o matriz) de tres columnas que contiene las coordenadas
              en el plano y el atributo del segundo punto del triangulo:
              puntoTri2 = [x2 y2 z2];
            - Vector (o matriz) de tres columnas que contiene las coordenadas
              en el plano y el atributo del tercer punto del triangulo:
              puntoTri3 = [x3 y3 z3];
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            En el caso de ser matrices, los cuatro argumentos anteriores han
            de tener el mismo numero de filas
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza de las coordenadas del punto de
                interpolacion. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada. Queda asi expresada
                  una matriz varianza-covarianza diagonal:
                  varCov=[Sxp^2 Syp^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y su correlacion
                  Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxp^2 Sxpyp Syp^2];
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza de las coordenadas del primer punto del
                triangulo. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y del atributo
                  Queda asi expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sx1^2 Sy1^2 Sz1^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y del atributo y su
                  correlacion. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sx1^2 Sx1y1 Sx1z1 Sy1^2 Sy1z1 Sz1^2];
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza de las coordenadas del segundo punto del
                triangulo. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y del atributo
                  Queda asi expresada una matriz varianza-covarianza diagonal:
```

```
                  varCov=[Sx2^2 Sy2^2 Sz2^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y del atributo y su
                  correlacion. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sx2^2 Sx2y2 Sx2z2 Sy2^2 Sy2z2 Sz2^2];
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza de las coordenadas del tercer punto del
                triangulo. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y del atributo
                  Queda asi expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sx3^2 Sy3^2 Sz3^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada y del atributo y su
                  correlacion. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sx3^2 Sx3y3 Sx3z3 Sy3^2 Sy3z3 Sz3^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian cuatro
              variables de valor -1, no se hace el calculo del error

Salidas:    - Atributo interpolado en las coordenadas pasadas (escalar o
              vector columna), en las mismas unidades que los atributos de los
              puntos del triangulo
            - Varianza del atributo interpolado (escalar o vector columna), en
              las mismas unidades (al cuadrado) que las del atributo
              En el caso de no haber pasado datos para el calculo del error
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [Z]=geo_InterpolaTriangulo(Interp,Tri1,Tri2,Tri3);
            [Z]=geo_InterpolaTriangulo(Interp,Tri1,Tri2,Tri3,-1,-1,-1,-1);
            [Z,varZ]=geo_InterpolaTriangulo(Interp,Tri1,Tri2,Tri3,
            varCovInterp,varCovTri1,varCovTri2,varCovTri3);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_AltitudMdeTin.m
                               - geo_InterpolaInvDistEspacio.m
                               - geo_InterpolaInvDistPlano.m
                               - geo_InterpolaTin.m

Nota: La funcion geo_InterpolaTriangulo no tiene en cuenta si el punto de
      interpolacion pasado esta o no contenido en el triangulo. El usuario ha
      de ser consciente de que una interpolacion fuera de los limites del
      triangulo, pese a ser matematicamente correcta, puede arrojar resultados
      practicos inservibles

Ultima revision: 18-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 10.11. Carpeta 11-delaunay

En esta carpeta se almacenan las funciones relacionadas con el cálculo de triangulaciones de Delaunay y la interpolación de datos en las mismas.

- **geo_CurvadoTin.m**:

```
Funcion:    geo_CurvadoTin.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula curvas de nivel en una red de triangulos irregulares (TIN)

Entradas:   - Matriz de Nx3 elementos que almacena las coordenadas de los
              puntos de la nube triangulada. Cada fila corresponde a un punto:
              - Col. 1: Coordenada X
              - Col. 2: Coordenada Y
              - Col. 3: Coordenada Z
            - Matriz de Tx3 elementos que contiene la triangulacion de
              Delaunay de la nube de puntos pasada. Cada fila describe cada
              uno de los T triangulos que la forman. Cada columna es la
              posicion en la matriz de la nube de puntos que corresponde a
              cada vertice del triangulo
            - Escalar o vector indicador de las curvas de nivel a crear. Tres
              posibilidades:
              - Escalar, N: Se crearan N curvas de nivel equiespaciadas entre
                            las cotas maxima y minima (incluidas) de la nube
                            de puntos
              - Vector fila de dos elementos iguales: Se creara una sola curva
                            de nivel de cota igual al valor de los elementos
                            del vector
              - Vector columna de N elementos: Se crearan N curvas de nivel a
                            las cotas indicadas. Si alguna de las cotas esta
                            por encima o por debajo de alguno de los valores
                            indicados no se tendra en cuenta

Salidas:    - Matriz de definicion de curvas de nivel. Esta matriz siempre
              tiene dos filas. La estructura por columnas es la siguiente:

              [cota1  x1 x2...xN  cota2  x1 x2...xM......cotaC  x1 x2...xP
               nodos1 y1 y2...yN  nodos2 y1 y2...yM......nodosC y1 y2...yP]

              - cota1, cota2,...cotaC: Cota de la linea de nivel
                                       correspondiente
              - nodos1, nodos2,...nodos1: Numero de nodos de la linea de nivel
                                          correspondiente
              - x1, x2,...xN: Coordenada X de cada nodo de la linea
              - y1, y2,...yN: Coordenada Y de cada nodo de la linea

Ejemplo:    curvas=geo_CurvadoTin(puntos,triangulacion,niveles);

Funciones utilizadas de GEO: - geo_Find.m
                             - geo_EsNatural.m
                             - geo_DesactivaControles.m
                             - geo_OrdenaMatriz.m
                             - geo_RestauraControles.m
```

```
Funciones relacionadas de GEO: - geo_Delaunay.m

Nota: geo_CurvadoTin es una modificacion de la funcion 'tricontour.m', escrita
      por Duane Hanselman y que se puede encontrar en
      http://www.mathworks.com/matlabcentral/fileexchange/
      loadFile.do?objectId=11040&objectType=File
      Las modificaciones que se han hecho con respecto al programa original
      eliminan uno de los argumentos de salida, capacitan a la funcion para
      calcular curvas de nivel cuya cota sea igual a la altura maxima y minima
      de la nube de puntos y optimizan la salida de la funcion en el sentido
      de que eliminan los nodos repetidos de las curvas

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_Delaunay.m**:

```
Funcion:    geo_Delaunay.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula una triangulacion de Delaunay para una nube de puntos

Entradas:   - Identificador del algoritmo a utilizar para el calculo de la
              triangulacion. Dos posibilidades:
              - Caracter 'Q' para utilizar la funcion 'delaunay' de GNU/Octave
                y MatLab que utiliza la biblioteca Qhull para el calculo de la
                triangulacion (ver Nota 1 al final de la ayuda)
              - Caracter 'T' para utilizar la biblioteca Triangle escrita por
                Jonathan Richard Shewchuk (ver la Nota 2 al final de la ayuda)
            - Matriz de Nx2 elementos que almacena las coordenadas en el plano
              de los puntos de la nube a triangular (ha de haber un minimo de
              4 puntos). Cada fila corresponde a un punto:
              - Col. 1: Coordenada X
              - Col. 2: Coordenada Y
            Argumentos opcionales (solo seran tenidos en cuenta si se trabaja
            con el algoritmo 'T'):
              - Cadena de texto con las opciones a pasar al programa de
                triangulacion. Cada caracter en la cadena sera una opcion de
                entre las siguientes:
                - 'i': Se utilizara el algoritmo incremental para generar la
                       triangulacion en lugar del de divide y venceras (divide
                       and conquer)
                - 'F': Se utilizara el algoritmo sweepline de Steven Fortune
                       para generar la triangulacion en lugar del de divide y
                       venceras (divide and conquer)
                - 'Q': Elimina la informacion de procesado que aparece en
                       pantalla mientras se crea la malla. Los mensajes de
                       error no los elimina
                En el caso de seleccionar a la vez las opciones 'i' y 'F' (sea
                en el orden que sea) se utilizara el algoritmo incremental
                Para mas informacion acerca de las opciones ver la pagina web
                de ayuda de la biblioteca Triangle (ver Nota 2 al final de la
                ayuda)
              - Matriz vacia o de Lx2 elementos que almacena los segmentos que
                forman el borde de la nube de puntos a triangular. Estos
                segmentos tambien funcionan como constrenimientos, cada uno de
```

```
                los cuales solo formara parte de un triangulo. Cada fila de
                esta matriz corresponde a un segmento:
                - Col. 1: Posicion en la matriz de puntos pasada del nodo
                          inicial de la linea
                - Col. 2: Posicion en la matriz de puntos pasada del nodo
                          final de la linea
                Si este argumento es una matriz vacia el programa de
                triangulacion generara un borde para la nube de puntos
                Todos los puntos de la nube y lineas de rotura que queden
                fuera de los limites definidos no seran tenidos en cuenta
                El limite definido ha de ser un poligono cerrado, ha de partir
                y llegar al mismo punto de la nube o no se generara
                triangulacion alguna
              - Matriz vacia o de Rx2 elementos que almacena las lineas de
                rotura o segmentos de constrenimiento a aplicar a la
                triangulacion. De esta forma se obliga a que los segmentos
                indicados sean lados de algun triangulo de la triangulacion
                Imponer estas lines implica que algunos triangulos pueden no
                respetar la condicion de Delaunay. Cada fila de esta matriz
                corresponde a un segmento:
                - Col. 1: Posicion en la matriz de puntos pasada del nodo
                          inicial de la linea
                - Col. 2: Posicion en la matriz de puntos pasada del nodo
                          final de la linea
                Si este argumento es una matriz vacia no se tendran en cuenta
                lineas de rotura
                Hay que evitar que dos o mas lineas de rotura se corten ya
                que dara lugar a un funcionamiento incorrecto del programa de
                triangulacion
                Hay que evitar que el borde y alguna linea de rotura se corten
                ya que dara lugar a un funcionamiento incorrecto del programa
                de triangulacion
              - H Matriz vacia o de Hx2 elementos que almacena los posible
                huecos que haya en la nube de puntos y que no es necesario
                triangular. Cada fila de esta matriz corresponde a un punto
                que esta dentro del hueco definido por lineas de rotura en la
                matriz correspondiente:
                - Col. 1: Coordenada X de un punto que esta dentro de un hueco
                - Col. 2: Coordenada Y de un punto que esta dentro de un hueco
                Si se introducen huecos cuyo punto identificador no esta
                dentro de un poligono definido por lineas de rotura no se
                creara ningun triangulo en la nube de puntos

Salidas:    - Matriz de Tx3 elementos que contiene la triangulacion de
              Delaunay de la nube de puntos pasada. Cada fila describe cada
              uno de los T triangulos que la forman. Cada columna es la
              posicion en la matriz de la nube de puntos que corresponde a
              cada vertice del triangulo

Ejemplo:    T=geo_Delaunay(algoritmo,puntos);
            T=geo_Delaunay(algoritmo,puntos,opciones);
            T=geo_Delaunay(algoritmo,puntos,opciones,limites,rotura,opciones);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_Find.m
                             - geo_EsNatural.m
```

```
Funciones relacionadas de GEO: - geo_AltitudMdeTin.m
                               - geo_InterpolaTin.m

Nota 1: Tanto GNU/Octave como MatLab utilizan la biblioteca Qhull para
        calcular triangulaciones de Delaunay. Esta biblioteca no calcula
        triangulaciones constrenidas. Para mas informacion ver la ayuda de la
        funcion 'delaunay' de GNU/Octave o MatLab. Tambien esta disponible la
        web de Qhull: http://www.qhull.org/

Nota 2: La biblioteca Triangle ha sido creada por Jonathan Richard Shewchuk e
        implementa un algoritmo mas rapido que el de Qhull para generar
        triangulaciones de Delaunay. Tambien es posible aplicar
        constrenimientos a la triangulacion. Esta biblioteca no se distribuye
        bajo la licencia GPL, sino que posee una propia. Para mas informacion
        visitar la web http://www.cs.cmu.edu/~quake/triangle.html

Ultima revision: 04-09-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 10.12. Carpeta 12-sistEcuaciones

En esta carpeta se almacenan las funciones para resolver sistemas de ecuaciones.

### 10.12.1. Carpeta 01-mmcc

En esta carpeta se almacenan las funciones para resolver sistemas por el método de mínimos cuadrados (MMCC).

- **geo_MediaCoordenadas.m**:

```
Funcion:    geo_MediaCoordenadas.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula las coordenadas por MMCC de un punto a partir de una serie
            de aproximaciones del mismo

Entradas:   - Matriz de coordenadas aproximadas, de tamano nxm, con la
              siguiente estructura:
              fijos=[ X11  X12  .  X1m
                      ........  .  ...
                      Xn1  Xn2  .  Xnm];
              Cada fila corresponde a un punto:
              - Col. 1: Primera componente de la posicion del punto
              - Col. 2: Primera componente de la posicion del punto
              - ..............................................
              - Col. m: M-esima componente de la posicion del punto
            Argumento opcional:
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos. Dos posibilidades:
                - Vector (o matriz) de m columnas, cada una de las cuales
                  contiene la varianza de cada componente coordenada del punto
                  Queda asi expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sx1^2 Sx2^2 . . . Sxm^2];
                - Vector (o matriz) de SUM[i=1;i=m]{i} columnas, cada una de
```

```
                las cuales contiene la varianza de cada componente de
                vector y sus correlaciones. Queda asi expresada (por filas)
                una matriz varianza-covarianza completa:
                varCov=[Sx1^2 Sx1x2 ... Sx2^2 Sx2x3 ... Sxm^2];
            Las varianzas han de ser pasadas en valor absoluto (no se
            controla internamente)
            Si, en vez de la matriz varianza-covarianza, se envia una
            variable de valor -1, se considera que las coordenadas de los
            puntos no tienen incertidumbre. En este caso, aunque en el
            proceso de ajuste se obtendrian unos parametros de precision, se
            considera que al usuario no le interesan y el valor retornado
            de la variable varianza sera -1
            En el caso de pasar matrices varianza-covarianza diagonales
            puede haber errores en el caso de que alguna componente de
            varianza sea cero o muy cercana a cero debido a que la matriz
            sera invertida para hallar la matriz de pesos. Este hecho no se
            comprueba internamente por lo que es responsabilidad del usuario
            el cuidar las posibles singularidades. En el caso de matrices
            varianza-covarianza completas el caso seria el mismo si se pasa
            alguna matriz que tenga una o mas filas o columnas formadas
            completamente por ceros

Salidas:    - Vector de coordenadas ajustadas por MMCC del punto de trabajo:
              punto=[ X1  X2 ... Xm];
              - Col. 1: Primera componente ajustada de la posicion del punto
              - Col. 2: Primera componente ajustada de la posicion del punto
              - ..........................................................
              - Col. m: M-esima componente ajustada de la posicion del punto
            - Matriz varianza-covarianza del punto ajustado (en formato
              vector), en las mismas unidades (al cuadrado para las varianzas)
              que las de las coordenadas de entrada. Si en la entrada se ha
              utilizado una matriz diagonal, la salida tambien sera diagonal,
              de la forma:
              varCov=[Sx1^2 Sx2^2 ... Sxm^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Sx1^2 Sx1x2 ... Sx2^2 Sx2x3 ... Sxm^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)
              aunque, en este caso, se podria haber estimado al calcular la
              media de las coordenadas

Ejemplo:    coorAjust=geo_MediaCoordenadas(coorAprox);
            [coorAjust,varianza]=geo_MediaCoordenadas(coorAprox,-1);
            [coorAjust,varianza]=geo_MediaCoordenadas(coorAprox,varCov);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_VectorMatrizSimInterna.m
                             - geo_MatrizSimVectorInterna.m

Funciones relacionadas de GEO: ninguna

Ultima revision: 20-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_MinCuadConstrEcObs.m**:

- **geo_MinCuadEcCond.m**:

- **geo_MinCuadEcObs.m**:

```
Funcion:    geo_MinCuadEcObs.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve un sistema de ecuaciones de observacion, A*x=t+v por
            minimos cuadrados

Entradas:   - Codigo identificador del algoritmo utilizado para la resolucion
              del sistema. Pueden ser las cadenas de texto:
              - 'INV': El problema se resuelve mediante la inversion ordinaria
                       del sistema de ecuaciones normales
              - 'PALU': El problema se resuelve mediante la factorizacion
                        PA=LU del sistema de ecuaciones normales
              - 'CHOL': El problema se resuelve mediante la factorizacion de
                        Cholesky del sistema de ecuaciones normales
              - 'QR': El problema se resuelve mediante la factorizacion A=QR
                      de la matriz de diseno
              - 'SVD': El problema se resuelve mediante la descomposicion en
                       valores singulares de la matriz de diseno
              Si la matriz de diseno del problema es singular, la solucion se
              basa en la obtencion de la matriz seudoinversa a partir de la
              descomposicion en valores singulares
            - Matriz de diseno del sistema
            - Termino independiente del sistema, vector columna
            Argumento opcional:
              - Matriz de pesos del problema, dos posibilidades:
                - Vector fila, que almacena una matriz de pesos diagonal
                - Matriz de pesos completa
              Si se omite este argumento o se pasa el valor -1, el problema se
              resuelve sin considerar pesos o, lo que es lo mismo, con P=I

Salidas:    - Estructura que contiene diversos datos del problema planteado y
              resuelto. Sus campos son:
              - datos.filasA: Numero de filas de la matriz de diseno
              - datos.rangoTeorico: Rango teorico de la matriz de diseno
                                    r = min(filA,colA)
              - datos.rangoReal: Rango real de la matriz de diseno
              - datos.condicionA: Numero de condicion de la matriz de diseno
                                  ponderada
              - datos.condicionN: Numero de condicion de la matriz de
                                  ecuaciones normales
              - datos.gradosLibertad: Grados de libertad del problema
              - datos.s02: Varianza de la referencia de peso unidad a
                           posteriori
            - Vector columna que almacena las incognitas del problema
            - Vector columna que almacena los residuos ajustados vAj=A*xAj-t
            - Vector columna que almacena el termino independiente ajustado
              tAj = A*xAj-vAj
            - Matriz cofactor a posteriori de las incognitas, Qxx
```

```
            - Matriz cofactor a posteriori de los residuos, Qvv
            - Matriz cofactor a posteriori del termino independiente, Qtt
            Si se llama a la funcion solo con 4 argumentos de salida no se
            realiza calculo interno de las matrices cofactor

Ejemplo:    [datos,x,v,t,Qxx,Qvv,Qtt]=geo_MinCuadEcObs('SVD',A,t);
            [datos,x,v,t,Qxx,Qvv,Qtt]=geo_MinCuadEcObs('SVD',A,t,P);
            [datos,x,v,t,Qxx,Qvv,Qtt]=geo_MinCuadEcObs('SVD',A,t,-1);
            [datos,x,v,t]=geo_MinCuadEcObs('SVD',A,t,P);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_Find.m
                             - geo_QuitaEspacios.m
                             - geo_ResTriInf.m
                             - geo_ResTriSup.m

Funciones relacionadas de GEO: - geo_MinCuadConstrEcObs.m
                               - geo_MinCuadEcCond.m
                               - geo_MinCuadModMixto.m

Nota: Para ahorrar multiplicaciones matriciales y poder trabajar con la matriz
      de diseno en los algoritmos QR y SVD se procede a una factorizacion
      previa de la matriz de pesos, tal y como se puede ver en [SB-LAGG-1997],
      pags. 400-403. De esta forma P=U'*U, con U triangular superior y Ap=U*A,
      tp=U*t. Queda, entonces, Ap*x=tp+vp y se puede trabajar como si fuese
      un sistema generico A*x=t. Los vectores de residuos ajustados y termino
      independiente ajustado de salida no son ponderados, es decir, para su
      calculo se ha utilizado la matriz A original, no Ap
      En el caso en que a la funcion se le pasasen la matriz Ap y el vector tp
      desde el principio no se le pasara matriz de pesos. De esta forma, el
      vector de incognitas de salida seria correcto, los vectores de residuos
      y termino independiente ajustados serian ponderados, la matriz cofactor
      de las incognitas seria correcta, pero las matrices cofactor de los
      residuos y terminos independientes serian incorrectas ya que para su
      calculo se necesita la matriz A original
      En el caso en que se pasase una matriz A cuadrada, la funcion trabajara
      como si fuese rectangular, es decir, se forma la matriz normal si asi lo
      necesita el algoritmo seleccionado. La solucion sera correcta, pero se
      haran operaciones de mas

Ultima revision: 18-03-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_MinCuadModMixto.m**:

- **geo_ResTriInf.m**:

```
Funcion:    geo_ResTriInf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve un sistema de ecuaciones, Ax=B, con la matriz A
            triangular inferior
```

```
Entradas:   - Matriz A, cuadrada y triangular inferior
            - Vector de terminos independientes B. Dos posibilidades:
              - Vector columna, con el mismo numero de filas que la matriz A
              - Matriz con el mismo numero de filas que la matriz A, donde
                cada columna sera considerada como un vector B distinto

Salidas:    - Incognitas del sistema. Dos posibilidades:
              - Vector columna con la solucion del sistema en el caso que el
                termino B del sistema haya sido un vector
              - Matriz en la que cada columna es un vector solucion del
                en el caso que el termino B del sistema haya sido una matriz

Ejemplo:    [x]=geo_ResTriInf(A,B);

Funciones utilizadas de GEO: geo_WarningOct.m

Funciones relacionadas de GEO: - geo_ResTriSup.m

Ultima revision: 24-02-08

Nota: La funcion no controla internamente si la matriz del sistema es
      triangular ni si es regular

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ResTriSup.m**:

```
Funcion:    geo_ResTriSup.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve un sistema de ecuaciones, Ax=B, con la matriz A
            triangular superior

Entradas:   - Matriz A, cuadrada y triangular superior
            - Vector de terminos independientes B. Dos posibilidades:
              - Vector columna, con el mismo numero de filas que la matriz A
              - Matriz con el mismo numero de filas que la matriz A, donde
                cada columna sera considerada como un vector B distinto

Salidas:    - Incognitas del sistema. Dos posibilidades:
              - Vector columna con la solucion del sistema en el caso que el
                termino B del sistema haya sido un vector
              - Matriz en la que cada columna es un vector solucion del
                en el caso que el termino B del sistema haya sido una matriz

Ejemplo:    [x]=geo_ResTriSup(A,B);

Funciones utilizadas de GEO: - geo_WarningOct.m

Funciones relacionadas de GEO: - geo_ResTriInf.m

Ultima revision: 24-02-08

Nota: La funcion no controla internamente si la matriz del sistema es
      triangular ni si es regular

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

### 10.12.2. Carpeta 02-metodosRobustos

En esta carpeta se almacenan las funciones para resolver sistemas utilizando métodos robustos.

## 10.13. Carpeta 13-dibujo

En esta carpeta se guardan las funciones relacionadas el dibujo de redes geodésicas, etc.

### 10.13.1. Carpeta 01-estructuraDibujo

En esta carpeta se almacenan las funciones relacionadas conla creación y manupulación de la estructura `dibujo`.

- **geo_CreaBloqueDibujo.m**:

```
Funcion:    geo_CreaBloqueDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade bloques a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Matriz de cuatro columnas. Las tres primeras son las coordenadas
              (x,y,z) de situacion de los bloques. La cuarta es el acimut de
              giro del bloque, en radianes
            - Nombre de los bloques a insertar, previamente definidos en la
              estructura dibujo. Si en esta lista se define un bloque que no
              existe en la estructura se produce un error y se sale de la
              funcion
            - Nombre de las capas donde ira cada bloque a crear. Si todos los
              bloques van a ir en la misma capa este argumento puede ser una
              cadena de texto o una geoLista; si se van a crear varias ha de
              ser una geoLista. Si en esta lista se define una capa que no
              existe en la estructura dibujo, se crea la nueva capa y se lanza
              un aviso

Salidas:    - Estructura 'dibujo' actualizada con los nuevos bloques

Ejemplo:    dibujo=geo_CreaBloqueDibujo(dibujo,coordenadas,nombresBloques,
            nombresCapas);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_ControlDominio.m
                             - geo_EsColorTipoGrosor.m
                             - geo_EsTexto.m
                             - geo_EsLista.m
                             - geo_EsListaCapas.m
                             - geo_RepiteEnLista.m
                             - geo_UneALista.m
                             - geo_DameElementoLista.m
                             - geo_ExisteEnLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
```

```
                                - geo_CreaElipseDibujo.m
                                - geo_CreaLineaDibujo.m
                                - geo_CreaPolilineaDibujo.m
                                - geo_CreaPuntoDibujo.m
                                - geo_CreaSimboloDibujo.m
                                - geo_CreaTextoDibujo.m

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaCapaDibujo.m**:

```
Funcion:    geo_CreaCapaDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade nuevas capas a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Nombre de la capa a crear. Si solo se va a crear una capa este
              argumento puede ser una cadena de texto o una geoLista; si se
              van a crear varias ha de ser una geoLista
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las capas. La primera columna corresponde al color, la segunda
              al tipo de linea y la tercera a la visibilidad de la capa:
              0/1 -> visible/invisible. Los identificadores de atributos se
              guardan en la estructura geoVarGlobal.dibujo...Dxf{i} (excepto
              el de visibilidad). Si este argumento tiene una sola fila se
              aplican los mismos atributos a todas las capas

Salidas:    - Estructura 'dibujo' actualizada con las nuevas capas

Ejemplo:    dibujo=geo_CreaCapaDibujo(dibujo,nombresCapas,atributos);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_EsLista.m
                             - geo_EsTexto.m
                             - geo_EsListaCapas.m
                             - geo_HayRepeticionesEnLista.m
                             - geo_Lista.m
                             - geo_EsColorTipoVis.m
                             - geo_LongitudLista.m
                             - geo_ExisteEnLista.m
                             - geo_DameElementoLista.m
                             - geo_UneALista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
                               - geo_CreaElipseDibujo.m
                               - geo_CreaLineaDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaPuntoDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_CreaTextoDibujo.m
```

```
Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaCircunferenciaDibujo.m**:

```
Funcion:    geo_CreaCircunferenciaDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade circunferencias a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Matriz de cuatro columnas. Las tres primeras corresponden a las
              coordenadas (x,y,z) del centro de cada circunferencia, la cuarta
              es el radio
            - Matriz de tres columnas que guarda los atributos de dibujo. La
              primera columna corresponde al color, la segunda al tipo de
              linea y la tercera al grosor de la linea. Los identificadores de
              atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todas las circunferencias
            - Nombre de las capas donde ira cada circunferencia a crear. Si
              todas las circunferencias van a ir en la misma capa este
              argumento puede ser una cadena de texto o una geoLista; si se
              van a crear varias ha de ser una geoLista. Si en esta lista se
              define una capa que no existe en la estructura dibujo, se crea
              la nueva capa y se lanza un aviso

Salidas:    - Estructura 'dibujo' actualizada con las nuevas circunferencias

Ejemplo:    dibujo=geo_CreaCircunferenciaDibujo(dibujo,coordenadas,atributos,
            nombresCapas);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_Find.m
                             - geo_EsColorTipoGrosor.m
                             - geo_EsTexto.m
                             - geo_EsLista.m
                             - geo_EsListaCapas.m
                             - geo_RepiteEnLista.m
                             - geo_UneALista.m
                             - geo_DameElementoLista.m
                             - geo_ExisteEnLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCapaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
                               - geo_CreaElipseDibujo.m
                               - geo_CreaLineaDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaPuntoDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_CreaTextoDibujo.m
```

```
Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaDefinicionBloqueDibujo.m**:

```
Funcion:    geo_CreaDefinicionBloqueDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade nuevas definiciones de bloque a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Nombre del bloque a crear. Si solo se va a crear uno este
              argumento puede ser una cadena de texto o una geoLista; si se
              van a crear varios bloques ha de ser una geoLista
            - Matriz de tres columnas,correspondientes a las coordenadas
              (x,y,z) de cada nodo de la polilinea de cada bloque. Cada
              polilinea estara separada en la matriz por una fila con el valor
              NaN
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las polilineas. La primera columna corresponde al color, la
              segunda al tipo de linea y la tercera al grosor de la linea. Los
              identificadores de atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todas las polilineas
            - Matriz de tres columnas, correspondientes a las coordenadas
              (x,y,z) de cada punto componente de cada bloque
            - Matriz de dos columnas que guarda los atributos de dibujo de los
              puntos. La primera columna corresponde al color y la segunda al
              grosor. Los identificadores de atributos se guardan en la
              estructura geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todos los puntos

Salidas:    - Estructura 'dibujo' actualizada con las nuevas definiciones de
              bloque

Ejemplo:    dibujo=geo_CreaDefinicionBloqueDibujo(dibujo,nombresBloques,
            coordPoli,atributosPoli,coordPuntos,atributosPuntos);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_EsLista.m
                             - geo_EsTexto.m
                             - geo_EsListaCapas.m
                             - geo_HayRepeticionesEnLista.m
                             - geo_Lista.m
                             - geo_Find.m
                             - geo_EsColorTipoGrosor.m
                             - geo_EsColorGrosor.m
                             - geo_ExisteEnLista.m
                             - geo_DameElementoLista.m
                             - geo_UneALista.m
                             - geo_RestauraControles.m
```

```
Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCapaDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaElipseDibujo.m
                               - geo_CreaLineaDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaPuntoDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_CreaTextoDibujo.m

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaElipseDibujo.m**:

```
Funcion:    geo_CreaElipseDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade elipses a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Matriz de seis columnas. Las tres primeras corresponden a las
              coordenadas (x,y,z) del centro de cada elipse, la cuarta es la
              longitud del semieje mayor, la quinta la longitud del semieje
              menor y la sexta la orientacion, esto es, el acimut del semieje
              mayor, en radianes
            - Matriz de tres columnas que guarda los atributos de dibujo. La
              primera columna corresponde al color, la segunda al tipo de
              linea y la tercera al grosor de la linea. Los identificadores de
              atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todas las circunferencias
            - Nombre de las capas donde ira cada elipse a crear. Si todas las
              elipses van a ir en la misma capa este argumento puede ser una
              cadena de texto o una geoLista; si se van a crear varias ha de
              ser una geoLista. Si en esta lista se define una capa que no
              existe en la estructura dibujo, se crea la nueva capa y se lanza
              un aviso

Salidas:    - Estructura 'dibujo' actualizada con las nuevas elipses

Ejemplo:    dibujo=geo_CreaElipseDibujo(dibujo,coordenadas,atributos,
            nombresCapas);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_Find.m
                             - geo_ControlDominio.m
                             - geo_EsColorTipoGrosor.m
                             - geo_EsTexto.m
                             - geo_EsLista.m
                             - geo_EsListaCapas.m
                             - geo_RepiteEnLista.m
                             - geo_UneALista.m
                             - geo_DameElementoLista.m
```

```
                              - geo_ExisteEnLista.m
                              - geo_DesactivaControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCapaDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
                               - geo_CreaLineaDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaPuntoDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_CreaTextoDibujo.m

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaLineaDibujo.m**:

```
Funcion:    geo_CreaLineaDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade lineas a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Matriz de seis columnas. Las tres primeras corresponden a las
              coordenadas (x,y,z) del punto inicio y las tres ultimas a las
              coordenadas (x,y,z) del punto final
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las lineas. La primera columna corresponde al color, la segunda
              al tipo de linea y la tercera al grosor de la linea. Los
              identificadores de atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todas las lineas
            - Nombre de las capas donde ira cada linea a crear. Si todas las
              lineas van a ir en la misma capa este argumento puede ser una
              cadena de texto o una geoLista; si se van a crear varias ha de
              ser una geoLista. Si en esta lista se define una capa que no
              existe en la estructura dibujo, se crea la nueva capa y se lanza
              un aviso

Salidas:    - Estructura 'dibujo' actualizada con las nuevas lineas

Ejemplo:    dibujo=geo_CreaLineaDibujo(dibujo,coorLineas,atriLineas,
            nombresCapas);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_EsColorTipoGrosor.m
                             - geo_EsTexto.m
                             - geo_EsLista.m
                             - geo_EsListaCapas.m
                             - geo_RepiteEnLista.m
                             - geo_UneALista.m
                             - geo_DameElementoLista.m
                             - geo_ExisteEnLista.m
```

```
                              - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCapaDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
                               - geo_CreaElipseDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaPuntoDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_CreaTextoDibujo.m

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaPolilineaDibujo.m**:

```
Funcion:    geo_CreaPolilineaDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade polilineas a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Matriz de tres columnas,correspondientes a las coordenadas
              (x,y,z) de cada nodo. Cada polilinea esta separada por una fila
              con el valor NaN
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las lineas. La primera columna corresponde al color, la segunda
              al tipo de linea y la tercera al grosor de la linea. Los
              identificadores de atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todas las lineas
            - Nombre de las capas donde ira cada linea a crear. Si todas las
              lineas van a ir en la misma capa este argumento puede ser una
              cadena de texto o una geoLista; si se van a crear varias ha de
              ser una geoLista. Si en esta lista se define una capa que no
              existe en la estructura dibujo, se crea la nueva capa y se lanza
              un aviso

Salidas:    - Estructura 'dibujo' actualizada con las nuevas polilineas

Ejemplo:    dibujo=geo_CreaPolilineaDibujo(dibujo,coorLineas,atriLineas,
            nombresCapas);

Nota: Aquellas polilineas que solo consten de un nodo y que hayan sido pasadas
      en la matriz con solo una fila de coordenadas permaneceran igual en la
      estructura

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_Find.m
                             - geo_EsColorTipoGrosor.m
                             - geo_EsTexto.m
                             - geo_EsLista.m
                             - geo_EsListaCapas.m
```

```
                                - geo_RepiteEnLista.m
                                - geo_UneALista.m
                                - geo_DameElementoLista.m
                                - geo_ExisteEnLista.m
                                - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                                - geo_CreaCapaDibujo.m
                                - geo_CreaCircunferenciaDibujo.m
                                - geo_CreaDefinicionBloqueDibujo.m
                                - geo_CreaElipseDibujo.m
                                - geo_CreaLineaDibujo.m
                                - geo_CreaPuntoDibujo.m
                                - geo_CreaSimboloDibujo.m
                                - geo_CreaTextoDibujo.m

Ultima revision: 02-09-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaPuntoDibujo.m**:

```
Funcion:    geo_CreaPuntoDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade puntos a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Matriz de tres columnas, correspondientes a las coordenadas
              (x,y,z) de cada punto a crear
            - Matriz de dos columnas que guarda los atributos de dibujo de los
              puntos. La primera columna corresponde al color y la segunda al
              grosor. Los identificadores de atributos se guardan en la
              estructura geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todos los puntos
            - Nombre de las capas donde ira cada punto a crear. Si todos los
              puntos van a ir en la misma capa este argumento puede ser una
              cadena de texto o una geoLista; si se van a crear varias ha de
              ser una geoLista. Si en esta lista se define una capa que no
              existe en la estructura dibujo, se crea la nueva capa y se lanza
              un aviso

Salidas:    - Estructura 'dibujo' actualizada con los nuevos puntos

Ejemplo:    dibujo=geo_CreaPuntoDibujo(dibujo,coorPuntos,atriPuntos,
            nombresCapas);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_EsColorGrosor.m
                             - geo_EsTexto.m
                             - geo_EsLista.m
                             - geo_EsListaCapas.m
                             - geo_RepiteEnLista.m
                             - geo_UneALista.m
                             - geo_DameElementoLista.m
```

```
                              - geo_ExisteEnLista.m
                              - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCapaDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
                               - geo_CreaElipseDibujo.m
                               - geo_CreaLineaDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_CreaTextoDibujo.m

Ultima revision: 02-09-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaSimboloDibujo.m**:

```
Funcion:    geo_CreaSimboloDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade simbolos a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Matriz de tres columnas,correspondientes a las coordenadas
              (x,y,z) de cada nodo de la polilinea que define al simbolo. Cada
              polilinea esta separada por una fila con el valor NaN
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las lineas. La primera columna corresponde al color, la segunda
              al tipo de linea y la tercera al grosor de la linea. Los
              identificadores de atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todas las lineas
            - Nombre de las capas donde ira cada simbolo a crear. Si todas los
              simbolos van a ir en la misma capa este argumento puede ser una
              cadena de texto o una geoLista; si se van a crear varias ha de
              ser una geoLista. Si en esta lista se define una capa que no
              existe en la estructura dibujo, se crea la nueva capa y se lanza
              un aviso

Salidas:    - Estructura 'dibujo' actualizada con los nuevos simbolos

Ejemplo:    dibujo=geo_CreaSimboloDibujo(dibujo,coorSimbolos,atriSimbolos,
            nombresCapas);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_Find.m
                             - geo_EsColorTipoGrosor.m
                             - geo_EsTexto.m
                             - geo_EsLista.m
                             - geo_EsListaCapas.m
                             - geo_RepiteEnLista.m
                             - geo_UneALista.m
                             - geo_DameElementoLista.m
```

```
                              - geo_ExisteEnLista.m
                              - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCapaDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
                               - geo_CreaElipseDibujo.m
                               - geo_CreaLineaDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaPuntoDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_CreaTextoDibujo.m

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaTextoDibujo.m**:

```
Funcion:    geo_CreaTextoDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Anade textos a una estructura 'dibujo'

Entradas:   - Una estructura 'dibujo'
            - Matriz de seis columnas. Las tres primeras corresponden a las
              coordenadas (x,y,z) del punto de anclaje de la parte izquierda
              del texto. La cuarta columna es la altura del texto, la quinta
              el codigo del punto de anclaje del texto, correspondiente con la
              estructura geoVarGlobal.dibujo.anclajeTextoDxf{i}. La sexta
              columna es el acimut de giro del texto, en radianes
            - Matriz de dos columnas que guarda los atributos de dibujo de los
              textos. La primera columna corresponde al color y la segunda al
              grosor. Los identificadores de atributos se guardan en la
              estructura geoVarGlobal.dibujo...Dxf{i}
              Si este argumento tiene una sola fila se aplican los mismos
              atributos a todos los textos
            - Textos a insertar, cadena de texo o geoLista
            - Nombre de las capas donde ira cada texto a crear. Si todos los
              textos van a ir en la misma capa este argumento puede ser una
              cadena de texto o una geoLista; si se van a crear varias ha de
              ser una geoLista. Si en esta lista se define una capa que no
              existe en la estructura dibujo, se crea la nueva capa y se lanza
              un aviso

Salidas:    - Estructura 'dibujo' actualizada con los nuevos textos

Ejemplo:    dibujo=geo_CreaTextoDibujo(dibujo,coordenadas,atributos,textos,
            nombresCapas);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_Find.m
                             - geo_EsNatural.m
                             - geo_ControlDominio.m
                             - geo_EsColorGrosor.m
```

```
                                - geo_EsTexto.m
                                - geo_EsLista.m
                                - geo_EsListaCapas.m
                                - geo_RepiteEnLista.m
                                - geo_UneALista.m
                                - geo_DameElementoLista.m
                                - geo_ExisteEnLista.m
                                - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                                - geo_CreaCapaDibujo.m
                                - geo_CreaCircunferenciaDibujo.m
                                - geo_CreaDefinicionBloqueDibujo.m
                                - geo_CreaElipseDibujo.m
                                - geo_CreaLineaDibujo.m
                                - geo_CreaPolilineaDibujo.m
                                - geo_CreaPuntoDibujo.m
                                - geo_CreaSimboloDibujo.m

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_Dibujo.m**:

```
Funcion:    geo_Dibujo.m

Proposito:  Crea una estructura 'dibujo' vacia

Entradas:   Ninguna

Salidas:    - Una estructura dibujo vacia, cuyos campos son:
              - dibujo.vacio = 1
              - dibujo.limites.hayLimVisDefinidos = 0
              - dibujo.limites.limitesVisualizacion = [0 0 0 0 0 0]
              - dibujo.capas.nombres = geo_Lista
              - dibujo.capas.atributos = zeros(0,3)
              - dibujo.defBloques = ''
              - dibujo.textos = ''
              - dibujo.puntos = ''
              - dibujo.bloques = ''
              - dibujo.circunferencias = ''
              - dibujo.elipses = ''
              - dibujo.simbolos = ''
              - dibujo.lineas = ''
              - dibujo.polilineas = ''

Ejemplo:    dibujo=geo_Dibujo;

Usa geoVarGlobal: No

Nota:       La estructura 'dibujo' completa es la siguiente:

      - dibujo.vacio: Escalar que indica si el dibujo esta vacio o no:
                           1/0 -> esta vacio/tiene entidades
      - dibujo.limites.limitesVisualizacion: Vector fila de seis columnas que
                           contiene las coordenadas de los limites de la
                           ventana de visualizacion del dibujo. La primera
```

```
                        columna contiene la coordenada X maxima, la
                        segunda la Y maxima, la tercera la Z maxima, la
                        cuarta la X minima, la quinta la Y minima y la
                        sexta la Z minima. Estas coordenadas limite no
                        son exactas sino aproximadas Para elementos del
                        dibujo como bloques o elipses se calcula una
                        cota superior de su tamano (sin tener en cuenta
                        su orientacion)
                .hayLimVisDefinidos: Variable que almacena la existencia
                        de limites de visualizacion definidos:
                        0/1 -> no hay/hay
- dibujo.capas.nombres: geoLista que contiene los nombres de las capas
                        del dibujo como cadenas de texto
              .atributos: Matriz de tres columnas que guarda los
                        atributos de dibujo de las capas. La primera
                        columna corresponde al color, la segunda al tipo
                        de linea y la tercera a la visibilidad de la
                        capa: 0/1 -> visible/invisible. Los
                        identificadores de atributos se guardan en la
                        estructura geoVarGlobal.dibujo...Dxf{i} (excepto
                        el de visibilidad)
- dibujo.lineas.coordenadas: Matriz de seis columnas. Las tres primeras
                        corresponden a las coordenadas (X,Y,Z) del punto
                        inicio y las tres ultimas a las coordenadas
                        (X,Y,Z) del punto final
               .capas: geoLista que contiene los nombres de las capas a
                        las que pertenecen las lineas
               .atributos: Matriz de tres columnas que guarda los
                        atributos de dibujo de las lineas. Tiene la
                        misma estructura que dibujo.capas.atributos
- dibujo.polilineas.coordenadas: Matriz de tres columnas,
                        correspondientes a las coordenadas (X,Y,Z) de
                        cada nodo. Cada polilinea esta separada por una
                        fila con el valor NaN
                   .capas: geoLista que contiene los nombres de las
                        capas a las que pertenecen las polilineas
                   .atributos: Matriz de tres columnas que guarda los
                        atributos de dibujo de las polilineas. Tiene la
                        misma estructura que dibujo.capas.atributos
- dibujo.puntos.coordenadas: Matriz de tres columnas, correspondientes a
                        las coordenadas (X,Y,Z) de cada punto
               .capas: geoLista que contiene los nombres de las capas a
                        las que pertenecen los puntos
               .atributos: Matriz de dos columnas que guarda los
                        atributos de dibujo de los puntos. La primera
                        columna corresponde al color y la segunda al
                        grosor. Los identificadores de atributos se
                        guardan en la estructura
                        geoVarGlobal.dibujo...Dxf{i}
- dibujo.simbolos: Tiene identicos campos que dibujo.polilineas. Cada
                        simbolo es, en realidad, una polilinea
- dibujo.circunferencias.coordenadas: Matriz de cuatro columnas. Las
                        tres primeras corresponden a las coordenadas
                        (X,Y,Z) del centro de cada circunferencia, la
                        cuarta es el radio
                     .capas: geoLista que contiene los nombres de las
                        capas a las que pertenecen las circunferencias
```

```
                    .atributos: Matriz de tres columnas que guarda los
                      atributos de dibujo de las circunferencias
                      Tiene la misma estructura que
                      dibujo.capas.atributos
- dibujo.elipses.coordenadas: Matriz de seis columnas. Las tres primeras
                      corresponden a las coordenadas (x,y,z) del
                      centro de cada elipse, la cuarta es la longitud
                      del semieje mayor, la quinta la longitud del
                      semieje menor y la sexta la orientacion, esto
                      es, el acimut del semieje mayor, en radianes
                .capas: geoLista que contiene los nombres de las capas a
                      las que pertenecen las elipses
                .atributos: Matriz de tres columnas que guarda los
                      atributos de dibujo de las elipses. Tiene la
                      misma estructura que dibujo.capas.atributos
- dibujo.textos.coordenadas: Matriz de seis columnas. Las tres primeras
                      corresponden a las coordenadas (x,y,z) del punto
                      de anclaje de la parte izquierda del texto. La
                      cuarta columna es la altura del texto, la quinta
                      el codigo del punto de anclaje del texto,
                      correspondiente con la estructura
                      geoVarGlobal.dibujo.anclajeTextoDxf{i}. La sexta
                      columna es el acimut de giro del texto, en
                      radianes
              .textos: geoLista que contiene los textos a escribir
              .capas: geoLista que contiene los nombres de las capas a
                      las que pertenecen los textos
              .atributos: Matriz de dos columnas que guarda los
                      atributos de dibujo de los textos. Tiene la
                      misma estructura que dibujo.puntos.atributos
- dibujo.defBloques.nombres: geoLista que contiene los nombres de los
                      bloques definidos en el dibujo. Los bloques
                      estaran compuestos por una polilinea y un punto
                      y su punto de anclaje es el (0,0,0), al que
                      estan referidos la polilinea y el punto que lo
                      componen
                 .coordPoli: Matriz de tres columnas que almacena las
                      coordenadas de los nodos de cada polilinea
                      Tiene la misma estructura que
                      dibujo.polilineas.coordenadas
                 .atriPoli: Matriz de tres columnas que guarda los
                      atributos de dibujo de las polilineas. Tiene la
                      misma estructura que dibujo.capas.atributos
                 .coordPuntos: Matriz de tres columnas que almacena
                      las coordenadas de los puntos.Tiene la misma
                      estructura que dibujo.puntos.coordenadas
                 .atriPuntos: Matriz de dos columnas que guarda los
                      atributos de dibujo de los puntos. Tiene la
                      misma estructura que dibujo.puntos.atributos
- dibujo.bloques.nombres: geoLista que contiene los nombres de los
                      bloques a dibujar
                .capas: geoLista que contiene los nombres de las capas a
                      las que pertenecen los bloques
                .coordenadas: Matriz de cuatro columnas. Las tres
                      primeras son las coordenadas (X,Y,Z) de
                      situacion de los bloques. La cuarta es el acimut
                      de giro del bloque, en radianes
```

```
Original:   29-08-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 09-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

- **geo_EsColorGrosor.m**:

```
Funcion:    geo_EsColorGrosor.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba si el argumento pasado es una matriz de definicion de
            los atributos color y grosor de una estructura dibujo

Entradas:   - Una variable
            Argumentos opcionales
              - Numero de filas que ha de tener la matriz, numero natural

Salidas:    - Numero de filas de la matriz, si el argumento de entrada es una
              matriz de atributos color, grosor de una estructura dibujo
            - Un error, en caso contrario

Ejemplo:    esAtri=geo_EsColorGrosor(atributos);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_EsNatural.m
                             - geo_Find.m

Funciones relacionadas de GEO: - geo_EsColorTipoGrosor.m
                               - geo_EsColorTipoVis.m
                               - geo_EsListaCapas.m

Ultima revision: 22-04-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_EsColorTipoGrosor.m**:

```
Funcion:    geo_EsColorTipoGrosor.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba si el argumento pasado es una matriz de definicion de
            los atributos color, tipo y grosor de una estructura dibujo

Entradas:   - Una variable
            Argumentos opcionales
              - Numero de filas que ha de tener la matriz, numero natural

Salidas:    - Numero de filas de la matriz, si el argumento de entrada es una
              matriz de atributos color, tipo, grosor de una estructura dibujo
            - Un error, en caso contrario

Ejemplo:    esAtri=geo_EsColorTipoGrosor(atributos);
```

```
Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_EsNatural.m
                             - geo_Find.m

Funciones relacionadas de GEO: - geo_EsColorGrosor.m
                               - geo_EsColorTipoVis.m
                               - geo_EsListaCapas.m

Ultima revision: 21-04-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_EsColorTipoVis.m**:

```
Funcion:    geo_EsColorTipoVis.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba si el argumento pasado es una matriz de definicion de
            los atributos color, tipo y visibilidad de una estructura dibujo

Entradas:   - Una variable
            Argumentos opcionales
              - Numero de filas que ha de tener la matriz, numero natural

Salidas:    - Numero de filas de la matriz, si el argumento de entrada es una
              matriz de atributos color, tipo, visibilidad de una estructura
              dibujo
            - Un error, en caso contrario

Ejemplo:    esAtri=geo_EsColorTipoVis(atributos);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_EsNatural.m
                             - geo_Find.m
                             - geo_EsEntero.m

Funciones relacionadas de GEO: - geo_EsColorGrosor.m
                               - geo_EsColorTipoGrosor.m
                               - geo_EsListaCapas.m

Ultima revision: 28-04-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_EsDibujo.m**:

```
Funcion:    geo_EsDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba si el argumento pasado es una estructura dibujo

Entradas:   - Una variable

Salidas:    - 1, si el argumento de entrada es una estructura dibujo
            - Un error, en caso contrario
```

```
Ejemplo:    esDibujo=geo_EsDibujo(dibujo);

Funciones utilizadas de GEO: - geo_DesactivaControles.m
                             - geo_EsEscalar.m
                             - geo_EsListaCapas.m
                             - geo_EsColorTipoVis.m
                             - geo_EsColorTipoGrosor.m
                             - geo_ControlDominio.m
                             - geo_EsColorGrosor.m
                             - geo_Find.m
                             - geo_EsNatural.m
                             - geo_ExisteEnLista.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCapaDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
                               - geo_CreaElipseDibujo.m
                               - geo_CreaLineaDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaPuntoDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_Dibujo.m
                               - geo_LimitesVisualDibujo.m

Ultima revision: 29-08-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_EsListaCapas.m**:

```
Funcion:    geo_EsListaCapas.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba si el argumento pasado es una geoLista con los nombres
            de las capas de una estructura dibujo

Entradas:   - Una variable
            Argumentos opcionales
              - Numero de elementos que ha de tener la geoLista, numero
                natural
              - geoLista con nombres de capas para evitar coincidencias

Salidas:    - Numero de elementos de la geoLista, si el argumento de entrada
              es una geoLista de una estructura dibujo
            - Un error, en caso contrario

Ejemplo:    esCapas=geo_EsListaCapas(capas);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_EsNatural.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_LongitudLista.m
```

```
                                - geo_EsTexto.m
                                - geo_DameElementoLista.m
                                - geo_ExisteEnLista.m
                                - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_EsColorGrosor.m
                                 - geo_EsColorTipoGrosor.m
                                 - geo_EsColorGrosorVis.m

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_LimitesVisualDibujo.m**:

```
Funcion:    geo_LimitesVisualDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula los limites de visulizacion de un dibujo

Entradas:   - Una estructura 'dibujo'

Salidas:    - Estructura 'dibujo' actualizada con los limites de visualizacion

Ejemplo:    dibujo=geo_LimitesVisualDibujo(dibujo);

Funciones utilizadas de GEO: - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_LongitudLista.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaBloqueDibujo.m
                               - geo_CreaCapaDibujo.m
                               - geo_CreaCircunferenciaDibujo.m
                               - geo_CreaDefinicionBloqueDibujo.m
                               - geo_CreaElipseDibujo.m
                               - geo_CreaLineaDibujo.m
                               - geo_CreaPolilineaDibujo.m
                               - geo_CreaPuntoDibujo.m
                               - geo_CreaSimboloDibujo.m
                               - geo_Dibujo.m
                               - geo_EsDibujo.m

Nota 1: Los limites calculados con esta funcion seran siempre algo mayores que
        los reales debido que se asumen ciertas generalizaciones para que la
        ejecucion de la funcion sea mas rapida. Por ejemplo, las elipses se
        consideran circunferencias de radio igual al semieje mayor. Todos los
        elementos complejos (textos, elipses, bloques) se asimilan a
        circunferencias con radio igual a su dimension mayor. De este modo,
        solo es necesario calcular los puntos extremos al norte, sur, este y
        oeste para calcular los limites; como contrapartida, estos seran
        estimados por exceso. Para una informacion mas detallada se recomienda
        ver el codigo de esta funcion

Nota 2: Si la estructura pasada ya tiene limites de visualizacion definidos se
        calculan de nuevo y se sobreescriben los antiguos
```

```
Nota 3: El calculo de los limites del dibujo tiene en cuenta todos los
        elementos definidos en la estructura dibujo, sean visibles o no

Ultima revision: 24-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

### 10.13.2. Carpeta 02-dibujoOctave

En esta carpeta se guardan las funciones relacionadas con el dibujo de redes geodésicas, elipses de error, etc. en *GNU Octave*.

- **geo_NodosCircunferencia.m**:

```
Funcion:    geo_NodosCircunferencia.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula las coordenadas de los nodos de la polilinea que
            representa una circunferencia y los devuelve en el formato
            adecuado para dibujarla con los comandos plot o plot3

Entradas:   - Coordenada X del centro geometrico de la circunferencia (escalar
              o vector columna)
            - Coordenada Y del centro geometrico de la circunferencia (escalar
              o vector columna)
            - Coordenada Z del centro geometrico de la circunferencia (escalar
              o vector columna)
            - Radio de la circunferencia (escalar o vector columna)
            Argumentos opcionales:
              - Cadena de texto que contendra el formato de las lineas
                dibujadas
              - Escalar indicador de las dimensiones del dibujo a presentar
                en pantalla (si ha lugar): 2/3 -> 2D/3D
                Por defecto este argumento vale 2

Salidas:    Si no se ha solicitado ningun argumento de salida:
              - Las circunferencias dibujadas en una ventana con el comando
                plot
            Si se ha solicitado un argumento de salida:
              - Matriz de tres columnas con las coordenadas de los nodos que
                forman cada circunferencia a dibujar en el formato adecuado
                para pasar al comando plot o plot3:
                - Col. 1: Coordenada X del nodo a dibujar
                - Col. 2: Coordenada Y del nodo a dibujar
                - Col. 3: Coordenada Z del nodo a dibujar
                Cada fila de la matriz es un nodo de una circunferencia. Entre
                circunferencia y circunferencia hay una fila con el indicador
                NaN que hace de division e impide que el comando plot dibuje
                todas las lineas que forman las circunferencias unidas:
                [x11   y11   z11
                 x12   y12   z12
                 x13   y13   z13
                      .     .
                 x1N   y1N   z1N
                 NaN   NaN   NaN
```

```
                    x21   y21   z21
                    x22   y22   z22
                    x23   y23   z23
                        .     .
                    x2N   y2N   z2N
                    NaN   NaN   NaN
                        .     .
                        .     .
                        .     .    ]
                    En el caso de haber pasado la linea de formato y/o el
                    indicador de dimensiones del dibujo, no se tendran en cuenta

  Ejemplo:    geo_NodosCircunferencia(xCentro,yCentro,zCentro,radio);
              geo_NodosCircunferencia(xCentro,yCentro,zCentro,radio,formato);
              nodos=geo_NodosCircunferencia(xCentro,yCentro,zCentro,radio);
              nodos=geo_NodosCircunferencia(xCentro,yCentro,zCentro,radio,
              formato);

  Funciones utilizadas de GEO: - geo_EsTexto.m
                               - geo_EsEscalar.m
                               - geo_Find.m
                               - geo_DesactivaControles.m
                               - geo_AjustaFilasVectorMatriz.m
                               - geo_QuitaLeyendaAtribDibujo.m
                               - geo_RestauraControles.m

  Funciones relacionadas de GEO: - geo_NodosCuadrado.m
                                 - geo_NodosCurvado.m
                                 - geo_NodosElipse.m
                                 - geo_NodosEstrella.m
                                 - geo_NodosLinea.m
                                 - geo_NodosTin.m
                                 - geo_NodosTriangEq.m

  Nota 1: Aunque se pasen coordenadas tridimensionales, las dimensiones de las
          circunferencias solo se aplicaran en el plano (x,y)

  Nota 2: El numero de nodos que se calcula para cada circunferencia viene
          definido en la variable global geoVarGlobal.dibujo.nodosCircunferencia

  Ultima revision: 25-02-08

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_NodosCuadrado.m**:

```
  Funcion:    geo_NodosCuadrado.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Calcula las coordenadas de las esquinas de un cuadrado y las
              devuelve en el formato adecuado para dibujarlo con los comandos
              plot o plot3

  Entradas:   - Coordenada X del centro geometrico del cuadrado (escalar o
                vector columna)
              - Coordenada Y del centro geometrico del cuadrado (escalar o
                vector columna)
```

```
              - Coordenada Z del centro geometrico del cuadrado (escalar o
                vector columna)
              - Longitud del lado del cuadrado (escalar o vector columna)
              - Acimut de giro del cuadrado, en radianes (escalar o vector
                columna)
              Argumentos opcionales:
                - Cadena de texto que contendra el formato de las lineas
                  dibujadas
                - Escalar indicador de las dimensiones del dibujo a presentar
                  en pantalla (si ha lugar): 2/3 -> 2D/3D
                  Por defecto este argumento vale 2

Salidas:      Si no se ha solicitado ningun argumento de salida:
                - Los cuadrados dibujados en una ventana con el comando plot
              Si se ha solicitado un argumento de salida:
                - Matriz de tres columnas con las coordenadas de los nodos que
                  forman cada cuadrado a dibujar en el formato adecuado para
                  pasar al comando plot o plot3:
                  - Col. 1: Coordenada X del nodo a dibujar
                  - Col. 2: Coordenada Y del nodo a dibujar
                  - Col. 3: Coordenada Z del nodo a dibujar
                  Cada fila de la matriz es un nodo de un cuadrado. Entre
                  cuadrado y cuadrado hay una fila con el indicador NaN que hace
                  de division e impide que el comando plot dibuje todas las
                  lineas que forman los cuadrados unidas:
                  [x11   y11   z11
                   x12   y12   z12
                   x13   y13   z13
                   x14   y14   z14
                   x11   y11   z11
                   NaN   NaN   NaN
                   x21   y21   z21
                   x22   y22   z22
                   x23   y23   z23
                   x24   y24   z24
                   x21   y21   z21
                   NaN   NaN   NaN
                       .     .
                       .     .
                       .     .    ]
                   En el caso de haber pasado la linea de formato y/o el
                   indicador de dimensiones del dibujo, no se tendran en cuenta

Ejemplo:      geo_NodosCuadrado(xCentro,yCentro,zCentro,lado,aci);
              geo_NodosCuadrado(xCentro,yCentro,zCentro,lado,aci,formato);
              nodos=geo_NodosCuadrado(xCentro,yCentro,zCentro,lado,aci);
              nodos=geo_NodosCuadrado(xCentro,yCentro,zCentro,lado,aci,formato);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_EsEscalar.m
                             - geo_ControlDominio.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_QuitaLeyendaAtribDibujo.m
                             - geo_RestauraControles.m
```

```
Funciones relacionadas de GEO: - geo_NodosCircunferencia.m
                               - geo_NodosCurvado.m
                               - geo_NodosElipse.m
                               - geo_NodosEstrella.m
                               - geo_NodosLinea.m
                               - geo_NodosTin.m
                               - geo_NodosTriangEq.m

Nota 1: Aunque se pasen coordenadas tridimensionales, las dimensiones y giros
        de los cuadrados solo se aplicaran en el plano (x,y)

Nota 2: Un cuadrado con acimut de giro igual a cero tendra el aspecto que se
        puede ver en la siguiente figura:
                                    ---
                                   |   |
                                   |___|

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_NodosCurvado.m**:

```
Funcion:    geo_NodosCurvado.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula las coordenadas de los nodos de las polilineas que
            representan curvas de nivel y los devuelve en el formato adecuado
            para dibujarlas con los comandos plot o plot3

Entradas:   - Matriz de definicion de curvas de nivel. Tiene la misma
              estructura que la matriz de salida de la funcion
              geo_CurvadoDelaunay.m
            Argumentos opcionales:
              - Cadena de texto que contendra el formato de las lineas
                dibujadas
              - Escalar indicador de las dimensiones del dibujo a presentar
                en pantalla (si ha lugar): 2/3 -> 2D/3D
                Por defecto este argumento vale 2

Salidas:    Si no se ha solicitado ningun argumento de salida:
              - Las curvas de nivel dibujadas en una ventana con el comando
                plot
            Si se ha solicitado un argumento de salida:
              - Matriz de tres columnas con las coordenadas de los puntos que
                forman cada curva de nivel a dibujar en el formato adecuado
                para pasar al comando plot o plot3:
                - Col. 1: Coordenada X del nodo a dibujar
                - Col. 2: Coordenada Y del nodo a dibujar
                - Col. 3: Coordenada Z del nodo a dibujar
                Cada fila de la matriz es un nodo de una curva. Entre curva y
                curva hay una fila con el indicador NaN que hace de division e
                impide que el comando plot dibuje todas las lineas de nivel
                unidas:
                [x11   y11   z11
                 x12   y12   z12
                     .     .
```

```
                    x1N   y1N   z1N
                    NaN   NaN   NaN
                    x21   y21   z21
                    x22   y22   z22
                        .     .
                    x2M   y2M   z2M
                    NaN   NaN   NaN
                        .     .
                        .     .
                        .     .    ]
                    En el caso de haber pasado la linea de formato y/o el
                    indicador de dimensiones del dibujo, no se tendran en cuenta

    Ejemplo:    geo_NodosCurvado(curvas);
                geo_NodosCurvado(curvas,formato);
                curvado=geo_NodosCurvado(curvas);
                curvado=geo_NodosCurvado(curvas,formato);

    Funciones utilizadas de GEO: - geo_EsTexto.m
                                 - geo_EsEscalar.m
                                 - geo_DesactivaControles.m
                                 - geo_QuitaLeyendaAtribDibujo.m
                                 - geo_RestauraControles.m

    Funciones relacionadas de GEO: - geo_NodosCircunferencia.m
                                   - geo_NodosCuadrado.m
                                   - geo_NodosElipse.m
                                   - geo_NodosEstrella.m
                                   - geo_NodosLinea.m
                                   - geo_NodosTin.m
                                   - geo_NodosTriangEq.m

    Ultima revision: 25-02-08

    Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_NodosElipse.m**:

```
    Funcion:    geo_NodosElipse.m

    Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

    Proposito:  Calcula las coordenadas de los nodos de la polilinea que
                representa una elipse y los devuelve en el formato adecuado para
                dibujarla con los comandos plot o plot3

    Entradas:   - Coordenada X del centro geometrico de la elipse (escalar o
                  vector columna)
                - Coordenada Y del centro geometrico de la elipse (escalar o
                  vector columna)
                - Coordenada Z del centro geometrico de la elipse (escalar o
                  vector columna)
                - Semieje mayor de la elipse (escalar o vector columna)
                - Semieje menor de la elipse (escalar o vector columna)
                - Acimut del semieje mayor de la elipse, en radianes (escalar o
                  vector columna)
                Argumentos opcionales:
                  - Cadena de texto que contendra el formato de las lineas
```

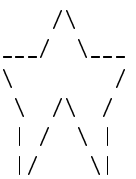
```
                dibujadas
              - Escalar indicador de las dimensiones del dibujo a presentar
                en pantalla (si ha lugar): 2/3 -> 2D/3D
                Por defecto este argumento vale 2

Salidas:    Si no se ha solicitado ningun argumento de salida:
              - Las elipses dibujadas en una ventana con el comando plot
            Si se ha solicitado un argumento de salida:
              - Matriz de tres columnas con las coordenadas de los nodos que
                forman cada elipse a dibujar en el formato adecuado para pasar
                al comando plot o plot3:
                - Col. 1: Coordenada X del nodo a dibujar
                - Col. 2: Coordenada Y del nodo a dibujar
                - Col. 3: Coordenada Z del nodo a dibujar
                Cada fila de la matriz es un nodo de una elipse. Entre elipse
                y elipse hay una fila con el indicador NaN que hace de
                division e impide que el comando plot dibuje todas las lineas
                que forman las elipses unidas:
                [x11   y11   z11
                 x12   y12   z12
                 x13   y13   z13
                     .     .
                 x1N   y1N   z1N
                 NaN   NaN   NaN
                 x21   y21   z21
                 x22   y22   z22
                 x23   y23   z23
                     .     .
                 x2N   y2N   z2N
                 NaN   NaN   NaN
                     .     .
                     .     .
                     .     .    ]
                 En el caso de haber pasado la linea de formato y/o el
                 indicador de dimensiones del dibujo, no se tendran en cuenta

Ejemplo:    geo_NodosElipse(xCentro,yCentro,zCentro,semiMayor,semiMenor,aci);
            geo_NodosElipse(xCentro,yCentro,zCentro,semiMayor,semiMenor,aci,
            formato);
            nodos=geo_NodosElipse(xCentro,yCentro,zCentro,semiMayor,semiMenor,
            aci);
            nodos=geo_NodosElipse(xCentro,yCentro,zCentro,semiMayor,semiMenor,
            aci,formato);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_EsEscalar.m
                             - geo_ControlDominio.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_AcimutAnguloMate.m
                             - geo_QuitaLeyendaAtribDibujo.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_NodosCircunferencia.m
                               - geo_NodosCuadrado.m
                               - geo_NodosCurvado.m
```

```
                                - geo_NodosEstrella.m
                                - geo_NodosLinea.m
                                - geo_NodosTin.m
                                - geo_NodosTriangEq.m

Nota 1: Aunque se pasen coordenadas tridimensionales, las dimensiones y giros
        de las elipses solo se aplicaran en el plano (x,y)

Nota 2: El numero de nodos que se calcula para cada elipse viene definido en
        la variable global geoVarGlobal.dibujo.nodosElipse

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_NodosEstrella.m**:

```
Funcion:    geo_NodosEstrella.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula las coordenadas de las esquinas de una estrella de cinco
            puntas y las devuelve en el formato adecuado para dibujarla con
            los comandos plot o plot3

Entradas:   - Coordenada X del centro geometrico de la estrella (escalar o
              vector columna)
            - Coordenada Y del centro geometrico de la estrella (escalar o
              vector columna)
            - Coordenada Z del centro geometrico de la estrella (escalar o
              vector columna)
            - Radio de la circunferencia que circunscribe a la estrella
              (escalar o vector columna)
            - Acimut de giro de la estrella, en radianes (escalar o vector
              columna)
            Argumentos opcionales:
              - Cadena de texto que contendra el formato de las lineas
                dibujadas
              - Escalar indicador de las dimensiones del dibujo a presentar
                en pantalla (si ha lugar): 2/3 -> 2D/3D
                Por defecto este argumento vale 2

Salidas:    Si no se ha solicitado ningun argumento de salida:
              - Las estrellas dibujadas en una ventana con el comando plot
            Si se ha solicitado un argumento de salida:
              - Matriz de tres columnas con las coordenadas de los nodos que
                forman cada estrella a dibujar en el formato adecuado para
                pasar al comando plot o plot3:
                - Col. 1: Coordenada X del nodo a dibujar
                - Col. 2: Coordenada Y del nodo a dibujar
                - Col. 3: Coordenada Z del nodo a dibujar
                Cada fila de la matriz es un nodo de una estrella. Entre
                estrella y estrella hay una fila con el indicador NaN que hace
                de division e impide que el comando plot dibuje todas las
                lineas que forman las estrellas unidas:
                [x11   y11   z11
                 x12   y12   z12
                 x13   y13   z13
```

```
                x14   y14   z14
                x15   y15   z15
                x11   y11   z11
                NaN   NaN   NaN
                x21   y21   z21
                x22   y22   z22
                x23   y23   z23
                x24   y24   z24
                x25   y25   z25
                x21   y21   z21
                NaN   NaN   NaN
                    .     .
                    .     .
                    .     .    ]
                En el caso de haber pasado la linea de formato y/o el
                indicador de dimensiones del dibujo, no se tendran en cuenta

Ejemplo:    geo_NodosEstrella(xCent,yCent,zCent,radio,aci);
            geo_NodosEstrella(xCent,yCent,zCent,radio,aci,formato);
            nodos=geo_NodosEstrella(xCent,yCent,zCent,radio,aci);
            nodos=geo_NodosEstrella(xCent,yCent,zCent,radio,aci,formato);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_EsEscalar.m
                             - geo_ControlDominio.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_QuitaLeyendaAtribDibujo.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_NodosCircunferencia.m
                               - geo_NodosCuadrado.m
                               - geo_NodosCurvado.m
                               - geo_NodosElipse.m
                               - geo_NodosLinea.m
                               - geo_NodosTin.m
                               - geo_NodosTriangEq.m

Nota 1: Aunque se pasen coordenadas tridimensionales, las dimensiones y giros
        de las estrellas solo se aplicaran en el plano (x,y)

Nota 2: Una estrella con acimut de giro igual a cero tendra el aspecto que se
        puede ver en la siguiente figura:
```



```
Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_NodosLinea.m**:

```
Funcion:    geo_NodosLinea.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula las coordenadas de los extremos de un segmento y las
            devuelve en el formato adecuado para dibujarlo con los comandos
            plot o plot3

Entradas:   - Coordenada X del punto inicial de la linea (escalar o vector
              columna)
            - Coordenada Y del punto inicial de la linea (escalar o vector
              columna)
            - Coordenada Z del punto inicial de la linea (escalar o vector
              columna)
            - Coordenada X del punto de llegada de la linea (escalar o vector
              columna)
            - Coordenada Y del punto de llegada de la linea (escalar o vector
              columna)
            - Coordenada Z del punto de llegada de la linea (escalar o vector
              columna)
            Argumentos opcionales:
              - Cadena de texto que contendra el formato de las lineas
                dibujadas
              - Escalar indicador de las dimensiones del dibujo a presentar
                en pantalla (si ha lugar): 2/3 -> 2D/3D
                Por defecto este argumento vale 2

Salidas:    Si no se ha solicitado ningun argumento de salida:
              - Las lineas dibujadas en una ventana con el comando plot
            Si se ha solicitado un argumento de salida:
              - Matriz de dos columnas con las coordenadas de los puntos que
                forman las lineas a dibujar en el formato adecuado para pasar
                al comando plot o plot3:
                - Col. 1: Coordenada X del nodo a dibujar
                - Col. 2: Coordenada Y del nodo a dibujar
                - Col. 3: Coordenada Z del nodo a dibujar
                Cada fila de la matriz es un nodo de la linea. Entre linea y
                linea hay una fila con el indicador NaN que hace de division e
                impide que el comando plot dibuje todas las lineas unidas:
                [x11   y11
                 x12   y12
                 NaN   NaN
                 x21   y21
                 x22   y22
                 NaN   NaN
                     .
                     .
                     .    ]
                 En el caso de haber pasado la linea de formato y/o el
                 indicador de dimensiones del dibujo, no se tendran en cuenta

Ejemplo:    geo_NodosLinea(x1,y1,z1,x2,y2,z2);
            geo_NodosLinea(x1,y1,z1,x2,y2,z2,formato);
            lineas=geo_NodosLinea(x1,y1,z1,x2,y2,z2);
            lineas=geo_NodosLinea(x1,y1,z1,x2,y2,z2,formato);

Funciones utilizadas de GEO: - geo_EsTexto.m
```

```
                                - geo_EsEscalar.m
                                - geo_DesactivaControles.m
                                - geo_AjustaFilasVectorMatriz.m
                                - geo_QuitaLeyendaAtribDibujo.m
                                - geo_RestauraControles.m

   Funciones relacionadas de GEO: - geo_NodosCircunferencia.m
                                  - geo_NodosCuadrado.m
                                  - geo_NodosCurvado.m
                                  - geo_NodosElipse.m
                                  - geo_NodosEstrella.m
                                  - geo_NodosTin.m
                                  - geo_NodosTriangEq.m

   Ultima revision: 03-05-08

   Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_NodosTin.m**:

```
   Funcion:    geo_NodosTin.m

   Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

   Proposito:  Calcula las coordenadas de los nodos de las polilineas que
               representan los triangulos de una triangulacion y los devuelve en
               el formato adecuado para dibujarlas con los comandos plot o plot3

   Entradas:   - Matriz de Nx3 elementos que almacena las coordenadas de los
                 puntos de la nube de triangulos. Cada fila corresponde a
                 un punto:
                 - Col. 1: Coordenada X
                 - Col. 2: Coordenada Y
                 - Col. 3: Coordenada Z
               - Matriz de Tx3 elementos que contiene la triangulacion de la nube
                 de puntos pasada. Cada fila describe cada uno de los T
                 triangulos que la forman. Cada columna es la posicion en la
                 matriz de la nube de puntos que corresponde a cada vertice del
                 triangulo
               Argumentos opcionales:
                 - Cadena de texto que contendra el formato de las lineas
                   dibujadas
                 - Escalar indicador de las dimensiones del dibujo a presentar
                   en pantalla (si ha lugar): 2/3 -> 2D/3D
                   Por defecto este argumento vale 2

   Salidas:    Si no se ha solicitado ningun argumento de salida:
                 - La triangulacion dibujada en una ventana con el comando plot
               Si se ha solicitado un argumento de salida:
                 - Matriz de tres columnas con las coordenadas de los puntos que
                   forman cada triangulo a dibujar en el formato adecuado para
                   pasar al comando plot o plot3:
                   - Col. 1: Coordenada X del vertice a dibujar
                   - Col. 2: Coordenada Y del vertice a dibujar
                   - Col. 3: Coordenada Z del vertice a dibujar
                   Cada fila de la matriz es un vertice de un triangulo. Entre
                   triangulo y triangulo hay una fila con el indicador NaN que
                   hace de division e impide que el comando plot dibuje todas las
```

```
                lineas que forman los triangulos unidas:
                [x11   y11   z11
                 x12   y12   z12
                 x13   y13   z13
                 x11   y11   z11
                 NaN   NaN   NaN
                 x21   y21   z21
                 x22   y22   z21
                 x23   y23   z21
                 x21   y21   z21
                 NaN   NaN   NaN
                     .     .
                     .     .
                     .     .    ]
                 En el caso de haber pasado la linea de formato y/o el
                 indicador de dimensiones del dibujo, no se tendran en cuenta

Ejemplo:    geo_NodosTin(puntos,triangulos);
            geo_NodosTin(puntos,triangulos,formato);
            triangulos=geo_NodosTin(puntos,triangulos);
            triangulos=geo_NodosTin(puntos,triangulos,formato);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_EsEscalar.m
                             - geo_Find.m
                             - geo_EsNatural.m
                             - geo_DesactivaControles.m
                             - geo_QuitaLeyendaAtribDibujo.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_NodosCircunferencia.m
                               - geo_NodosCuadrado.m
                               - geo_NodosCurvado.m
                               - geo_NodosElipse.m
                               - geo_NodosEstrella.m
                               - geo_NodosLinea.m
                               - geo_NodosTriangEq.m

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_NodosTriangEq.m**:

```
Funcion:    geo_NodosTriangEq.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula las coordenadas de las esquinas de un triangulo equilatero
            y las devuelve en el formato adecuado para dibujarlo con los
            comandos plot o plot3

Entradas:   - Coordenada X del centro geometrico del triangulo (escalar o
              vector columna)
            - Coordenada Y del centro geometrico del triangulo (escalar o
              vector columna)
            - Coordenada Z del centro geometrico del triangulo (escalar o
              vector columna)
```

```
              - Radio de la circunferencia que circunscribe al triangulo
                (escalar o vector columna)
              - Acimut de giro del triangulo, en radianes (escalar o vector
                columna)
              Argumentos opcionales:
                - Cadena de texto que contendra el formato de las lineas
                  dibujadas
                - Escalar indicador de las dimensiones del dibujo a presentar
                  en pantalla (si ha lugar): 2/3 -> 2D/3D
                  Por defecto este argumento vale 2

Salidas:      Si no se ha solicitado ningun argumento de salida:
                - Los triangulos dibujados en una ventana con el comando plot
              Si se ha solicitado un argumento de salida:
                - Matriz de tres columnas con las coordenadas de los nodos que
                  forman cada triangulo a dibujar en el formato adecuado para
                  pasar al comando plot o plot3:
                  - Col. 1: Coordenada X del nodo a dibujar
                  - Col. 2: Coordenada Y del nodo a dibujar
                  - Col. 3: Coordenada Z del nodo a dibujar
                  Cada fila de la matriz es un nodo de un triangulo. Entre
                  triangulo y triangulo hay una fila con el indicador NaN que
                  hace de division e impide que el comando plot dibuje todas las
                  lineas que forman los triangulos unidas:
                  [x11   y11   z11
                   x12   y12   z12
                   x13   y13   z13
                   x11   y11   z11
                   NaN   NaN   NaN
                   x21   y21   z21
                   x22   y22   z22
                   x23   y23   z23
                   x21   y21   z21
                   NaN   NaN   NaN
                       .     .
                       .     .
                       .     .    ]
                  En el caso de haber pasado la linea de formato, no se tendra
                  en cuenta

Ejemplo:      geo_NodosTriangEq(xCent,yCent,zCent,radio,aci);
              geo_NodosTriangEq(xCent,yCent,zCent,radio,aci,formato);
              nodos=geo_NodosTriangEq(xCent,yCent,zCent,radio,aci);
              nodos=geo_NodosTriangEq(xCent,yCent,zCent,radio,aci,formato);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_ControlDominio.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_QuitaLeyendaAtribDibujo.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_NodosCircunferencia.m
                               - geo_NodosCuadrado.m
                               - geo_NodosCurvado.m
                               - geo_NodosElipse.m
```

```
                                - geo_NodosEstrella.m
                                - geo_NodosLinea.m
                                - geo_NodosTin.m

Nota 1: Aunque se pasen coordenadas tridimensionales, las dimensiones y giros
        de los triangulos solo se aplicaran en el plano (x,y)

Nota 2: Un triangulo con acimut de giro igual a cero tendra el aspecto que se
        puede ver en la siguiente figura:

                                    /\
                                   /__\

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_QuitaLeyendaAtribDibujo.m**:

```
Funcion:    geo_QuitaLeyendaAtribDibujo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Elimina la cadena de texto identificativa de la leyenda de la
            cadena de atributos del dibujo pasada al comando plot. Esta cadena
            de texto no existe en el comando plot de MatLab, pero si en
            GNU/Octave. La cadena de titulo va indicada entre ; -> ';Titulo;'

Entradas:   - Cadena de texto que contendra el formato de un dibujo
            Argumento opcional:
              - Escalar de valor 1 o 0 que indica si la leyenda se ha de
                eliminar forzosamente o solo si estamos trabajando en MatLab
                respectivamente
              Si se omite este argumento la leyenda se eliminara solo si
              estamos trabajando en MatLab

Salidas:    - La cadena de atributos introducida sin el atributo de texto de
              leyenda en el caso correspondiente
            - La cadena de texto identificadora de la leyenda sin los simbolos
              identificadores ';' aunque no se haya quitado de la cadena de
              atributos

Ejemplo:    cadena=geo_QuitaLeyendaAtribDibujo(cadena);

Funciones utilizadas de GEO: - geo_EsTexto.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 31-08-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

### 10.13.3. Carpeta 03-dibujoDxf

En esta carpeta se guardan las funciones relacionadas con el dibujo de redes geodésicas, elipses de error, etc. en ficheros DXF.

- **geo_CreaCapaDxf.m**:

```
Funcion:    geo_CreaCapaDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Define una nueva capa de dibujo en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - geoLista con los nombre de las capas a escribir, como cadenas de
              texto. La funcion no controla internamente esta ultima propiedad
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las capas. La primera columna corresponde al color, la segunda
              al tipo de linea y la tercera a la visibilidad de la capa:
              0/1 -> visible/invisible. Los identificadores de atributos se
              guardan en la estructura geoVarGlobal.dibujo...Dxf{i} (excepto
              el de visibilidad). La funcion no controla internamente si estos
              codigos son correctos. Esta matriz ha de tener el mismo numero
              de filas que la geoLista de nombres

Salidas:    - Numero de capas escritas si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario

Ejemplo:    capas = geo_CreaCapaDxf(idFichero,nombres,atributos);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_LongitudLista.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.m
                               - geo_DibujaCircunferenciaDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaDefinicionBloqueDxf.m**:

```
Funcion:    geo_CreaDefinicionBloqueDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Crea nuevas definiciones de bloque en un fichero DXF
```

```
Entradas:   - Identificador de fichero abierto para escribir
            - geoLista con los nombre de los bloques a definir, como cadenas
              de texto. La funcion no controla internamente esta ultima
              propiedad
            - Matriz de tres columnas,correspondientes a las coordenadas
              (x,y,z) de cada nodo de la polilinea de cada bloque. Cada
              polilinea estara separada en la matriz por una fila con el valor
              NaN. El numero de polilineas definidas en esta matriz ha de ser
              igual al numero de bloques definido en la geoLista de entrada
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las polilineas. La primera columna corresponde al color, la
              segunda al tipo de linea y la tercera al grosor de la linea. Los
              identificadores de atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}. La funcion no controla
              internamente si estos codigos son correctos. Esta matriz ha de
              tener el mismo numero de filas que la geoLista de nombres
            - Matriz de tres columnas, correspondientes a las coordenadas
              (x,y,z) de cada punto componente de cada bloque. El numero de
              puntos definidos en esta matriz ha de ser igual al numero de
              bloques definido en la geoLista de entrada. Este hecho no se
              controla internamente
            - Matriz de dos columnas que guarda los atributos de dibujo de los
              puntos. La primera columna corresponde al color y la segunda al
              grosor. Los identificadores de atributos se guardan en la
              estructura geoVarGlobal.dibujo...Dxf{i}. La funcion no controla
              internamente si estos codigos son correctos. Esta matriz ha de
              tener el mismo numero de filas que la geoLista de nombres

Salidas:    - Numero de bloques definidos si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario

Ejemplo:    numeroBloques = geo_CreaDefinicionBloqueDxf(idFichero,nombres,
            coordPoli,atriPoli,coordPuntos,atriPuntos);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_Find.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea
```

```
Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaDibujoDxf.m**:

```
Funcion:    geo_CreaDibujoDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Crea un fichero DXF a partir de una estructura 'dibujo'

Entradas:   - Nombre del fichero a crear, como cadena de texto
            - Estructura 'dibujo'

Salidas:    - 1, si el dibujo se ha creado correctamente
            - Un fichero (en el directorio de trabajo) en formato DXF v.11/12
              cuyo nombre es la cadena de texto pasada como primer argumento

Ejemplo:    estado = geo_CreaDibujoDxf(nombreFichero,estructuraDibujo);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_EsDibujo.m
                             - geo_DesactivaControles.m
                             - geo_CreaTipoLineaDxf.m
                             - geo_LongitudLista.m
                             - geo_CreaCapaDxf.m
                             - geo_CreaDefinicionBloqueDxf.m
                             - geo_DibujaPolilineaDxf.m
                             - geo_DibujaLineaDxf.m
                             - geo_DibujaCircunferenciaDxf.m
                             - geo_DibujaElipseDxf.m
                             - geo_DibujaSimboloDxf.m
                             - geo_DibujaBloqueDxf.m
                             - geo_DibujaPuntoDxf.m
                             - geo_DibujaTextoDxf.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.
                               - geo_DibujaCirculoDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Toda la informacion necesaria sobre el formato DXF ha sido extraida de:
      http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=8446698

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaHeaderDxf.m**:

```
Funcion:    geo_CreaHeaderDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Define la seccion HEADER de un fichero DXF version 12

Entradas:   - Identificador de fichero abierto para escribir
            Argumentos opcionales:
              - Vector fila de seis columnas que contiene las coordenadas de
                los limites de la ventana de visualizacion del dibujo. La
                primera columna contiene la coordenada X maxima, la segunda la
                Y maxima, la tercera la Z maxima, la cuarta la X minima, la
                quinta la Y minima y la sexta la z minima
              - Variable que almacena si se aplican o no los limites
                establecidos en el vector pasado como segundo argumento:
                0/1 -> no se aplican/se aplican

Salidas:    - 1, si la funcion se ha ejecutado correctamente
            - Un error en caso contrario

Ejemplo:    estado = geo_CreaHeaderDxf(idFichero);
            estado = geo_CreaHeaderDxf(idFichero,limites,aplicaLimites);

Funciones utilizadas de GEO: - geo_EsFichero.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.m
                               - geo_DibujaCircunferenciaDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota 1: Esta funcion comieza a escribir en el lugar donde este el puntero de
        L/E del fichero pasado y finaliza con un retorno de carro y nueva
        linea

Nota 2: Las definiciones escritas son las siguientes:
        - Version: AutoCAD 12
        - Direccion del origen de angulos: este (origen matematico)
        - Sentido de medida de angulos: antihorario
        - Unidades angulares: grados sexagesimales decimales
        - Sin rejilla
        - Sin snap

Ultima revision: 01-05-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_CreaTipoLineaDxf.m**:

```
Funcion:    geo_CreaTipoLineaDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Define los tipos de linea en un fichero DXF. Esta funcion define
            los tipos CONTINUOUS, DASHED, DASHDOT y DOT

Entradas:   - Identificador de fichero abierto para escribir

Salidas:    - 1, si la funcion se ha ejecutado correctamente
            - Un error en caso contrario

Ejemplo:    estado = geo_CreaTipoLineaDxf(idFichero);

Funciones utilizadas de GEO: - geo_EsFichero.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_DibujaBloqueDxf.m
                               - geo_DibujaCircunferenciaDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 29-04-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DibujaBloqueDxf.m**:

```
Funcion:    geo_DibujaBloqueDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Dibuja un bloque previamente defiido en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - Matriz de cuatro columnas. Las tres primeras son las coordenadas
              (x,y,z) de situacion de los bloques. La cuarta es el acimut de
              giro del bloque, en radianes
            - geoLista con los nombres de los bloques a insertar
            - geoLista con los nombre de las capas donde ira cada bloque, como
              cadenas de texto. La funcion no controla internamente esta
              ultima propiedad. Ha de tener el mismo numero de elementos que
              bloques a escribir

Salidas:    - Numero de bloques escritos si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario
```

```
Ejemplo:    bloques = geo_DibujaBloqueDxf(idFichero,coordenadas,bloques,
            capas);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_Find.m
                             - geo_AcimutAnguloMate.m
                             - geo_RadGdec.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaCirculoDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DibujaCircunferenciaDxf.m**:

```
Funcion:    geo_DibujaCircunferenciaDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Dibuja una circunferencia en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - Matriz de cuatro columnas. Las tres primeras corresponden a las
              coordenadas (x,y,z) del centro de cada circunferencia, la cuarta
              es el radio
            - Matriz de tres columnas que guarda los atributos de dibujo. La
              primera columna corresponde al color, la segunda al tipo de
              linea y la tercera al grosor de la linea. Los identificadores de
              atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}. La funcion no controla
              internamente si estos codigos son correctos
              Esta matriz ha de tener el mismo numero de filas que la de
              coordenadas
            - geoLista con los nombre de las capas donde ira cada
              circunferencia, como cadenas de texto. La funcion no controla
              internamente esta ultima propiedad. Ha de tener el mismo numero
              de elementos que circunferencias a escribir

Salidas:    - Numero de circunferencias escritas si la funcion se ha ejecutado
```

```
              correctamente
            - Un error en caso contrario

Ejemplo:    circunf = geo_DibujaCircunferenciaDxf(idFichero,coordenadas,
            atributos,capas);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DibujaElipseDxf.m**:

```
Funcion:    geo_DibujaElipseDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Dibuja una elipse como una polilinea en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - Matriz de seis columnas. Las tres primeras corresponden a las
              coordenadas (x,y,z) del centro de cada elipse, la cuarta es la
              longitud del semieje mayor, la quinta la longitud del semieje
              menor y la sexta la orientacion, esto es, el acimut del semieje
              mayor, en radianes
            - Matriz de tres columnas que guarda los atributos de dibujo. La
              primera columna corresponde al color, la segunda al tipo de
              linea y la tercera al grosor de la linea. Los identificadores de
              atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}. La funcion no controla
              internamente si estos codigos son correctos
              Esta matriz ha de tener el mismo numero de filas que la de
              coordenadas
            - geoLista con los nombre de las capas donde ira cada elipse, como
              cadenas de texto. La funcion no controla internamente esta
              ultima propiedad. Ha de tener el mismo numero de elementos que
              elipses a escribir
```

```
Salidas:    - Numero de elipses escritas si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario

Ejemplo:    elipses = geo_DibujaElipseDxf(idFichero,coordenadas,atributos,
            capas);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_NodosElipse.m
                             - geo_DibujaPolilineaDxf.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.
                               - geo_DibujaCirculoDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 29-04-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DibujaLineaDxf.m**:

```
Funcion:    geo_DibujaLineaDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Dibuja una linea en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - Matriz de seis columnas. Las tres primeras corresponden a las
              coordenadas (x,y,z) del punto inicio y las tres ultimas a las
              coordenadas (x,y,z) del punto final
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las lineas. La primera columna corresponde al color, la segunda
              al tipo de linea y la tercera al grosor de la linea. Los
              identificadores de atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}. La funcion no controla
              internamente si estos codigos son correctos
              Esta matriz ha de tener el mismo numero de filas que la de
              coordenadas
            - geoLista con los nombre de las capas donde ira cada linea, como
              cadenas de texto. La funcion no controla internamente esta
              ultima propiedad. Ha de tener el mismo numero de elementos que
```

```
              lineas a escribir

Salidas:    - Numero de lineas escritas si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario

Ejemplo:    lineas = geo_DibujaLineaDxf(idFichero,coordenadas,atributos,
            capas);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.m
                               - geo_DibujaCircunferenciaDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DibujaPolilineaDxf.m**:

```
Funcion:    geo_DibujaPolilineaDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Dibuja una polilinea en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - Matriz de tres columnas,correspondientes a las coordenadas
              (x,y,z) de cada nodo. Cada polilinea esta separada por una fila
              con el valor NaN
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las lineas. La primera columna corresponde al color, la segunda
              al tipo de linea y la tercera al grosor de la linea. Los
              identificadores de atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}. La funcion no controla
              internamente si estos codigos son correctos
              Esta matriz ha de tener el mismo numero de filas que la de
              coordenadas
            - geoLista con los nombre de las capas donde ira cada linea, como
              cadenas de texto. La funcion no controla internamente esta
              ultima propiedad. Ha de tener el mismo numero de elementos que
```

```
              lineas a escribir

Salidas:    - Numero de polilineas escritas si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario

Ejemplo:    lineas = geo_DibujaPolilineaDxf(idFichero,coordenadas,atributos,
            capas);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_Find.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.m
                               - geo_DibujaCircunferenciaDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DibujaPuntoDxf.m**:

```
Funcion:    geo_DibujaPuntoDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Dibuja un punto en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - Matriz de tres columnas, correspondientes a las coordenadas
              (x,y,z) de cada punto a crear
            - Matriz de dos columnas que guarda los atributos de dibujo de
              cada punto. La primera columna corresponde al color y la segunda
              al grosor de la linea. Los identificadores de atributos se
              guardan en la estructura geoVarGlobal.dibujo...Dxf{i}. La
              funcion no controla internamente si estos codigos son correctos
              Esta matriz ha de tener el mismo numero de filas que la de
              coordenadas
            - geoLista con los nombre de las capas donde ira cada punto, como
              cadenas de texto. La funcion no controla internamente esta
              ultima propiedad. Ha de tener el mismo numero de elementos que
              puntos a escribir
```

```
Salidas:    - Numero de puntos escritos si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario

Ejemplo:    puntos = geo_DibujaPuntoDxf(idFichero,coordenadas,atributos,
            capas);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.m
                               - geo_DibujaCircunferenciaDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaSimboloDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DibujaSimboloDxf.m**:

```
Funcion:    geo_DibujaSimboloDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Dibuja un simbolo en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - Matriz de tres columnas,correspondientes a las coordenadas
              (x,y,z) de cada nodo de la polilinea que define al simbolo. Cada
              polilinea esta separada por una fila con el valor NaN
            - Matriz de tres columnas que guarda los atributos de dibujo de
              las lineas. La primera columna corresponde al color, la segunda
              al tipo de linea y la tercera al grosor de la linea. Los
              identificadores de atributos se guardan en la estructura
              geoVarGlobal.dibujo...Dxf{i}. La funcion no controla
              internamente si estos codigos son correctos
              Esta matriz ha de tener el mismo numero de filas que la de
              coordenadas
            - geoLista con los nombre de las capas donde ira cada simbolo,
              como cadenas de texto. La funcion no controla internamente esta
              ultima propiedad. Ha de tener el mismo numero de elementos que
              simbolos a escribir
```

```
Salidas:    - Numero de simbolos escritos si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario

Ejemplo:    simbolos = geo_DibujaSimboloDxf(idFichero,coordenadas,atributos,
            capas);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_DibujaPolilineaDxf.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.m
                               - geo_DibujaCircunferenciaDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaTextoDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DibujaTextoDxf.m**:

```
Funcion:    geo_DibujaTextoDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Dibuja texto en un fichero DXF

Entradas:   - Identificador de fichero abierto para escribir
            - Matriz de seis columnas. Las tres primeras corresponden a las
              coordenadas (x,y,z) del punto de anclaje de la parte izquierda
              del texto. La cuarta columna es la altura del texto, la quinta
              el codigo del punto de anclaje del texto, correspondiente con la
              estructura geoVarGlobal.dibujo.anclajeTextoDxf{i}. La sexta
              columna es el acimut de giro del texto, en radianes
            - Matriz de dos columnas que guarda los atributos de dibujo de los
              textos. La primera columna corresponde al color y la segunda al
              grosor. Los identificadores de atributos se guardan en la
              estructura geoVarGlobal.dibujo...Dxf{i}. La funcion no controla
              internamente si estos codigos son correctos
              Esta matriz ha de tener el mismo numero de filas que la de
              coordenadas
            - geoLista con los textos a insertar
            - geoLista con los nombre de las capas donde ira cada texto, como
```

```
                cadenas de texto. La funcion no controla internamente esta
                ultima propiedad. Ha de tener el mismo numero de elementos que
                textos a escribir

Salidas:    - Numero de textos escritos si la funcion se ha ejecutado
              correctamente
            - Un error en caso contrario

Ejemplo:    elipses = geo_DibujaTextoDxf(idFichero,coordenadas,atributos,
            textos,capas);

Funciones utilizadas de GEO: - geo_EsFichero.m
                             - geo_EsLista.m
                             - geo_DesactivaControles.m
                             - geo_AcimutAnguloMate.m
                             - geo_RadGdec.m
                             - geo_DameElementoLista.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CreaCapaDxf.m
                               - geo_CreaDefinicionBloqueDxf.m
                               - geo_CreaDibujoDxf.m
                               - geo_CreaHeaderDxf.m
                               - geo_CreaTipoLineaDxf.m
                               - geo_DibujaBloqueDxf.
                               - geo_DibujaCirculoDxf.m
                               - geo_DibujaElipseDxf.m
                               - geo_DibujaLineaDxf.m
                               - geo_DibujaPolilineaDxf.m
                               - geo_DibujaPuntoDxf.m
                               - geo_DibujaSimboloDxf.m

Nota: Esta funcion comieza a escribir en el lugar donde este el puntero de L/E
      del fichero pasado y finaliza con un retorno de carro y nueva linea

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 10.14. Carpeta 14-funcionesCpp

En esta carpeta se almacenan los ficheros fuente y ejecutables de las funciones escritas en C++ de GEO.

### 10.14.1. Carpeta 01-fuentes

En esta carpeta se almacenan los ficheros fuente de la parte de GEO escrita en C++.

#### 10.14.1.1. Carpeta 01-fuentesGeneral

En esta carpeta se almacenan los ficheros fuente de las utilidades programadas en C++ para apoyo de las funciones de GEO escritas en dicho lenguaje.

**10.14.1.1.1. Carpeta** 01-h En esta carpeta se guardan los archivos de cabecera (*.h) de las utilidades de apoyo para la programación de las funciones de GEO escritas en C++.

- **constantesOct.h**:

```
Archivo: constantesOct.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Definicion de constantes a utilizar en la programacion de partes
           de GEO en C++

Ultima revision: 30-09-06

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **datosBinariosOct.h**:

```
Archivo: datosBinariosOct.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Declaracion de funciones de trabajo con datos binarios almacenados
           en ficheros a utilizar en la programacion de partes de GEO en C++

Ultima revision: 08-03-07

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **defineTriangle.h**:

```
Archivo: defineTriangle.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Definicion de la variable REAL para el correcto funcionamiento de
           las funciones que utilizan la biblioteca Triangle de Jonathan
           Richard Shewchuk para la generacion de triangulaciones de Delaunay
           http://www.cs.cmu.edu/~quake/triangle.html
           Para el correcto funcionamiento de esta biblioteca en GEO, REAL
           siempre ha de valer lo mismo tanto en este fichero como en
           triangle.c

Ultima revision: 20-08-07

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **funcionesAuxiliaresOct.h**:

```
Archivo: funcionesAuxiliaresOct.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Declaracion de funciones auxiliares a utilizar en la programacion
           de partes de GEO en C++

Ultima revision: 12-08-08

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geodesiaOct.h**:

```
Archivo: geodesiaOct.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Declaracion de funciones para calculos geodesicos a utilizar en la
           programacion de partes de GEO en C++

Ultima revision: 11-03-07

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **gravimetriaOct.h**:

```
Archivo: gravimetriaOct.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Declaracion de funciones relacionadas con calculos gravimetricos a
           utilizar en la programacion de partes de GEO en C++

Ultima revision: 23-05-08

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **matematicasOct.h**:

```
Archivo: matematicasOct.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Declaracion de funciones matematicas a utilizar en la programacion
           de partes de GEO en C++

Ultima revision: 09-10-06

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **rutaLibOct.h**:

```
Archivo: rutaLibOct.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Declaracion de la ruta de las librerias C++ de Octave

Ultima revision: 01-10-06

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **triangle.h**:

```
Archivo: triangle.h

Escrito por: Jonathan Richard Shewchuk, jrs@cs.berkeley.edu

Proposito: Declaracion de funciones de la biblioteca Triangle. A continuacion
           se transcribe la licencia de la biblioteca:
```

```
These programs may be freely redistributed under the condition that the
copyright notices (including the copy of this notice in the code comments
and the copyright notice printed when the '-h' switch is selected) are
not removed, and no compensation is received.  Private, research, and
institutional use is free.  You may distribute modified versions of this
code UNDER THE CONDITION THAT THIS CODE AND ANY MODIFICATIONS MADE TO IT
IN THE SAME FILE REMAIN UNDER COPYRIGHT OF THE ORIGINAL AUTHOR, BOTH
SOURCE AND OBJECT CODE ARE MADE FREELY AVAILABLE WITHOUT CHARGE, AND
CLEAR NOTICE IS GIVEN OF THE MODIFICATIONS.  Distribution of this code as
part of a commercial system is permissible ONLY BY DIRECT ARRANGEMENT
WITH THE AUTHOR.  (If you are not directly supplying this code to a
customer, and you are instead telling them how they can obtain it for
free, then you are not required to make any arrangement with me.)

Informacion: http://www.cs.cmu.edu/~quake/triangle.html

Ultima revision: 28-07-05

Revisado por: Jonathan Richard Shewchuk, jrs@cs.berkeley.edu
```

- **triangleOct.h**:

```
Archivo: triangleOct.h

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Declaracion de funciones intermedias para utilizar la biblioteca
           Triangle de Jonathan Richard Shewchuk para la generacion de
           triangulaciones de Delaunay
           http://www.cs.cmu.edu/~quake/triangle.html

Ultima revision: 04-09-07

Revisado por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

**10.14.1.1.2. Carpeta** 02-cc En esta carpeta se guardan los ficheros fuente en C/C++ de las utilidades de apoyo para la programación de las funciones de enlace dinámico de GEO.

- **datosBinariosOct.cc**:

```
Archivo: datosBinariosOct.cc

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Definicion de funciones de trabajo con datos binarios almacenados
           en ficheros a utilizar en la programacion de partes de GEO en C++

Ultima revision: 16-12-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **funcionesAuxiliaresOct.cc**:

```
Archivo: funcionesAuxiliaresOct.cc

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
Proposito: Definicion de funciones auxiliares a utilizar en la programacion
           de partes de GEO en C++

Ultima revision: 15-08-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geodesiaOct.cc**:

```
Archivo: geodesiaOct.cc

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Definicion de funciones para calculos geodesicos a utilizar en la
           programacion de partes de GEO en C++

Ultima revision: 25-11-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **gravimetriaOct.cc**:

```
Archivo: gravimetriaOct.cc

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Definicion de funciones relacionadas con calculos gravimetricos a
           utilizar en la programacion de partes de GEO en C++

Ultima revision: 15-12-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **matematicasOct.cc**:

```
Archivo: matematicasOct.cc

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Definicion de funciones matematicas a utilizar en la programacion
           de partes de GEO en C++

Ultima revision: 09-10-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **triangle.c**:

```
Archivo: triangle.c

Escrito por: Jonathan Richard Shewchuk, jrs@cs.berkeley.edu

Proposito: Definicion de funciones de la biblioteca Triangle. A continuacion
           se transcribe la licencia de la biblioteca:

These programs may be freely redistributed under the condition that the
```

```
copyright notices (including the copy of this notice in the code comments
and the copyright notice printed when the '-h' switch is selected) are
not removed, and no compensation is received.  Private, research, and
institutional use is free.  You may distribute modified versions of this
code UNDER THE CONDITION THAT THIS CODE AND ANY MODIFICATIONS MADE TO IT
IN THE SAME FILE REMAIN UNDER COPYRIGHT OF THE ORIGINAL AUTHOR, BOTH
SOURCE AND OBJECT CODE ARE MADE FREELY AVAILABLE WITHOUT CHARGE, AND
CLEAR NOTICE IS GIVEN OF THE MODIFICATIONS.  Distribution of this code as
part of a commercial system is permissible ONLY BY DIRECT ARRANGEMENT
WITH THE AUTHOR.  (If you are not directly supplying this code to a
customer, and you are instead telling them how they can obtain it for
free, then you are not required to make any arrangement with me.)

Informacion: http://www.cs.cmu.edu/~quake/triangle.html

Nota 1: Para el correcto funcionamiento de esta biblioteca en GEO, REAL
        siempre ha de valer lo mismo tanto en este fichero como en
        defineTriangle.h

Nota 2: Se ha redefinido la variable INEXACT a INEXACT volatile para que
        Triangle funcione correctamente en GEO. Ver las lineas 296-318 de
        este fichero

Ultima revision: 28-07-05

Revisado por: Jonathan Richard Shewchuk, jrs@cs.berkeley.edu
```

- **triangleOct.cc**:

```
Archivo: triangleOct.c

Escrito por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito: Definicion de funciones intermedias para utilizar la biblioteca
           Triangle de Jonathan Richard Shewchuk para la generacion de
           triangulaciones de Delaunay
           http://www.cs.cmu.edu/~quake/triangle.html

Ultima revision: 04-09-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

#### 10.14.1.2. Carpeta 02-fuentesOct

En esta carpeta se almacenan los ficheros fuente de las funciones de GEO escritas en C++.

- **geo_AsciiBinOct.cc**:

```
Funcion:    geo_AsciiBinOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Convierte un fichero de formato ASCII a formato binario

Entradas:   - Nombre del fichero de texto original
            - Nombre del fichero binario de salida. Este fichero se creara
              sin ningun tipo de cabecera, comenzara directamente con los
              datos y estos estaran almacenados uno a continuacion de otro,
```

```
                sin bytes en blanco entre ellos
              - Nombre del tipo de dato de C/C++ a guardar en el fichero
                binario. Los tipos de dato permitidos estan definidos en la
                estructura global geoVarGlobal.nombreDatoBinC{i}
              - Tamano de trabajo en GEO del tipo de dato de C/C++ a guardar en
                el fichero binario. Debido a las diferentes implementaciones de
                C/C++, los tamanos de algunos tipos de dato pueden no coincidir
                con los definidos en nuestro compilador
              - Numero de lineas de la cabecera del fichero original que no
                contienen datos. Las lineas de cabecera no pueden tener mas de
                250 caracteres
              - Factor de multiplicacion para pasar los datos del fichero
                original a otra unidad cualquiera
              - Codigo del formato de escritura de los datos binarios en el
                fichero destino. Pueden ser tres codigos, segun el formato:
                - LITTLE-ENDIAN
                - BIG-ENDIAN
                - NATIVO: Tipo nativo de la maquina donde estamos trabajando
                          En este caso no se hace cambio de formato alguno
                Estos indicadores siempre han de ser pasados en mayusculas y
                sin contener ningun espacio en blanco

  Salidas:    - Fichero binario con los valores del fichero de texto. Los datos
                siguen manteniendo el orden de almacenamiento del fichero
                original. Se almacena en el directorio de trabajo
              - Numero de puntos convertidos

  Ejemplo:    [nPuntos]=geo_AsciiBinOct(fOrig,fSal,tDatoC,tamDatoC,lineasFuera,
              factorEscala,formatoBinario);

  Funciones relacionadas de GEO: Ninguna

  Ultima revision: 11-02-07

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ExtraeDatosFicheroMallaOct.cc**:

```
  Funcion:    geo_ExtraeDatosFicheroMallaOct.cc

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Extrae datos de un fichero binario en formato GEO

  Entradas:   - Tamano de la ventana de interpolacion, 1 o numero par
              - Indicador de cobertura global en latitud: 0/1->no/si
              - Indicador de cobertura global en longitud: 0/1->no/si
              - Numero de nodos de la malla en latitud
              - Numero de nodos de la malla en longitud
              - Vector columna que almacena la coordenada X del nodo anterior
                en longitud al punto de trabajo con respecto a la esquina
                noroeste del cuadrado que define la malla de puntos dato. La
                coordenada X se mide en la direccion del paralelo
              - Vector columna que almacena la coordenada Y del nodo anterior
                en latitud al punto de trabajo con respecto a la esquina
                noroeste del cuadrado que define la malla de puntos dato. La
                coordenada Y se mide en la direccion del meridiano
              - Nombre del tipo de dato en C/C++ guardado en el fichero binario
```

```
                - Longitud del tipo de dato guardado en el fichero binario
                - Codigo del formato de escritura de los datos binarios en el
                  fichero destino. Pueden ser tres codigos, segun el formato:
                  - LITTLE-ENDIAN
                  - BIG-ENDIAN
                  - NATIVO: Tipo nativo de la maquina donde estamos trabajando
                            En este caso no se hace cambio de formato alguno
                  Estos indicadores siempre han de ser pasados en mayusculas y
                  sin contener ningun espacio en blanco
                - Ruta completa del fichero binario que contiene los datos

    Salidas:    - Matriz de tamano (VENTANAxN)x(VENTANA) que contiene el valor
                  del dato del fichero para cada ventana de interpolacion

    Ejemplo:    matrizVentana=geo_ExtraeDatosFicheroMallaOct(ventana,cobGlobLat,
                cobGlobLon,puntosLat,puntosLon,preLat,preLon,nombreDato,
                longitudDato,tipoAlmac,nombreFichero);

    Funciones relacionadas de GEO: Ninguna

    Ultima revision: 03-03-07

    Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_FuncAsocLegendreSoloOct.cc**:

```
    Funcion:    geo_FuncAsocLegendreSoloOct.cc

    Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

    Proposito:  Calcula el valor de una funcion asociada de Legendre de grado N
                y orden M
                La formulacion (formulas de recurrencia) ha sido extraida de
                [TO-GEOD-2001], pag. 71 y [AS-UB91-1992], pag. 25 y desarrollada
                a partir de [YU-MMAT-2006], pag. 106

    Entradas:   - Grado de la funcion a calcular, numero entero >= 0
                - Orden de la funcion a calcular, numero entero <= grado
                - Argumento de la funcion (en el caso que nos ocupa, coseno de la
                  colatitud esferica del punto de trabajo)
                - Cadena de texto 'UN' o 'FN' o 'QN', segun se quieran calcular
                  las funciones sin normalizar (UNNORMALIZED), fuertemente
                  normalizadas (FULLY-NORMALIZAED) en el sentido de
                  [HM-GEFI-1985], pag. 23 o cuasi-normalizadas (QUASI-NORMALIZED)
                  en el sentido de [AS-UB91-1992], pag. 25

    Salidas:    - Escalar que contiene el valor de la funcion asociada de
                  Legendre indicada en la entrada para el punto de trabajo

    Ejemplo:    [fAsoc]=geo_FuncAsocLegendreSoloOct(grado,orden,t,'UN');
                [fAsoc]=geo_FuncAsocLegendreSoloOct(grado,orden,t,'FN');
                [fAsoc]=geo_FuncAsocLegendreSoloOct(grado,orden,t,'QN');

    Funciones utilizadas de GEO: Ninguna

    Funciones relacionadas de GEO:

    Ultima revision: 17-10-06
```

```
Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_FuncAsocLegendreTodoOct.cc**:

```
Funcion:    geo_FuncAsocLegendreSoloOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula los valores de una funcion asociada de Legendre de grado
            N y orden M
            La formulacion (formulas de recurrencia) ha sido extraida de
            [TO-GEOD-2001], pag. 71 y [AS-UB91-1992], pag. 25 y desarrollada
            a partir de [YU-MMAT-2006], pag. 106

Entradas:   - Grado de la funcion a calcular, numero entero >= 0
            - Orden de la funcion a calcular, numero entero <= grado
            - Argumento de la funcion (en el caso que nos ocupa, coseno de la
              colatitud esferica del punto de trabajo)
            - Cadena de texto 'UN' o 'FN' o 'QN', segun se quieran calcular
              las funciones sin normalizar (UNNORMALIZED), fuertemente
              normalizadas (FULLY-NORMALIZAED) en el sentido de
              [HM-GEFI-1985], pag. 23 o cuasi-normalizadas (QUASI-NORMALIZED)
              en el sentido de [AS-UB91-1992], pag. 25

Salidas:    - Matriz de (grado+1xorden+1) que contiene el valor de las
              funciones asociadas de Legendre de grado y orden <= a los
              pasados como argumentos
              fAL = [ F00
                      F10  F11
                      F20  F21  F22
                     ......................
                      FN0  FN1  FN2 ... FNM];
              De esta forma, el elemento (i,j) de la matriz corresponde al
              valor de la funcion asociada de grado i-1 y orden j-1
              Los elementos de la matriz que corresponden a funciones
              imposibles (orden > grado) contienen el valor 0

Ejemplo:    [fAsoc]=geo_FuncAsocLegendreTodoOct(grado,orden,t,'UN');
            [fAsoc]=geo_FuncAsocLegendreTodoOct(grado,orden,t,'FN');
            [fAsoc]=geo_FuncAsocLegendreTodoOct(grado,orden,t,'QN');

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO:

Ultima revision: 22-10-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GravedadPrismaOct.cc**:

```
Funcion:    geo_GravedadPrismaOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la componente vertical (eje Z) de la atraccion
```

```
            gravitatoria de un conjunto de prismas sobre un conjunto de
            puntos dados en el exterior o sobre su superficie
            La formulacion ha sido extraida de [NA-GRPR-2000]

  Entradas: - Vector fila o matriz de tres columnas con las coordenadas (en
              metros) de los puntos atraidos en un sistema cartesiano
              dextrogiro. Cada fila de la matriz es un punto distinto:
              - Col. 1: Coordenada X del punto
              - Col. 2: Coordenada Y del punto
              - Col. 3: Coordenada Z del punto
            - Vector fila o matriz de seis columnas con las coordenadas (en
              metros) de los limites de las caras del prisma atrayente en un
              sistema cartesiano dextrogiro. Cada fila de la matriz es un
              prisma distinto:
              - Col. 1: Coordenada X1 del prisma|
              - Col. 2: Coordenada X2 del prisma| Con (x1,y1,z1)<(x2,y2,z2)
              - Col. 3: Coordenada Y1 del prisma| respectivamente
              - Col. 4: Coordenada Y2 del prisma| Esta condicion no se
              - Col. 5: Coordenada Z1 del prisma| controla internamente
              - Col. 6: Coordenada Z2 del prisma|
            - Escalar o vector columna que almacena la densidad de cada
              prisma (en kg/m^3). Si es un vector ha de tener el mismo numero
              de filas que la matriz de prismas
            - Constante de gravitacion universal
            - Cadena de texto que almacena el identificador del metodo
              utilizado para el calculo de la atraccion gravitaroria. Las
              cadenas permitidas son 'NAGY' o 'MACMILLAN' segun se desee el
              metodo de Nagy o la aproximacion de MacMillan

  Salidas:  - Escalar o vector columna con la gravedad en cada punto debida a
              la atraccion de cada prisma. Si solo se ha introducido un punto
              atraido las filas del vector columna corresponden a la
              atraccion debida a cada prisma, mientras que si se han
              introducido varios puntos y varios prismas el vector columna
              contiene la atraccion debida a cada prisma sobre cada punto:
              [Pr1P1,...,PrNP1,Pr1P2,...PrNP2,...,Pr1PM,...,PrNPM]^T
              donde Pr->Prisma y P->Punto

  Ejemplo:  [grav]=geo_GravedadPrismaOct(puntos,prismas,densidades,G,metodo);

  Funciones utilizadas de GEO: Ninguna

  Funciones relacionadas de GEO: Ninguna

  Ultima revision: 25-03-08

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GravedadRectanguloOct.cc**:

```
  Funcion:  geo_GravedadRectanguloOct.cc

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito: Calcula la componente vertical (eje Z) de la atraccion
             gravitatoria de un conjunto de rectangulos contenidos en el plano
             XZ sobre un conjunto de puntos (tambien en el plano XZ) dados en
             el exterior o sobre su superficie (sobre uno de sus lados,
```

```
            excepto los vertices) segun la formulacion de [TE-APGE-1900]

Entradas:   - Vector fila o matriz de dos columnas con las coordenadas (en
              metros) de los puntos atraidos en un sistema cartesiano. Cada
              fila de la matriz es un punto distinto:
              - Col. 1: Coordenada X del punto
              - Col. 2: Coordenada Z del punto
            - Vector fila o matriz de cuatro columnas con las coordenadas (en
              metros) de los limites de las caras del rectangulo atrayente en
              un sistema cartesiano. Cada fila de la matriz es un rectangulo
              distinto:
              - Col. 1: Coordenada X1 del rectangulo| Con (x1,z1)<(x2,z2)
              - Col. 2: Coordenada X2 del rectangulo| respectivamente
              - Col. 3: Coordenada Z1 del rectangulo| Esta condicion no se
              - Col. 4: Coordenada Z2 del rectangulo| controla internamente
            - Escalar o vector columna que almacena la densidad de cada
              rectangulo (en kg/m^3). Si es un vector ha de tener el mismo
              numero de filas que la matriz de rectangulos
            - Constante de gravitacion universal

Salidas:    - Escalar o vector columna con la gravedad en cada punto debida a
              la atraccion de cada rectangulo. Si solo se ha introducido un
              punto atraido las filas del vector columna corresponden a la
              atraccion debida a cada rectangulo, mientras que si se han
              introducido varios puntos y varios rectangulos el vector
              columna contiene la atraccion debida a cada rectangulo sobre
              cada punto:
              [Re1P1,...,ReNP1,Re1P2,...ReNP2,...,Re1PM,...,ReNPM]^T
              donde Re->Rectangulo y P->Punto

Ejemplo:    [grav]=geo_GravedadRectanguloOct(puntos,rectangulos,densidades,
            G);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_GravedadPrismaOct.cc

Ultima revision: 23-03-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_InterpolaMallaOct.cc**:

```
Funcion:    geo_InterpolaMallaOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Realiza una interpolacion por el metodo del vecino mas proximo,
            bilineal o spline cubica natural en una matriz de datos

Entradas:   - Matriz de tamano (VENTANAxN)x(VENTANA) que contiene el valor
              del dato del fichero para cada ventana de interpolacion
            - Vector columna que almacena la coordenada X del punto de
              trabajo con respecto a la esquina sudoeste de la ventana de
              interpolacion. La coordenada X se mide en la direccion del
              paralelo
            - Vector columna que almacena la coordenada Y del punto de
              trabajo con respecto a la esquina sudoeste de la ventana de
```

```
                interpolacion. La coordenada Y se mide en la direccion del
                meridiano
              - Tamano de la ventana de interpolacion, 1 o numero par. Si el
                numero es 1 se utiliza el vecino mas proximo como solucion, si
                es 2 se realiza una interpolacion bilineal. En otro caso, la
                interpolacion se realiza por medio de splines cubicos

Salidas:      - Vector columna con los valores interpolados, en las mismas
                unidades que las de los valores dato de la matriz de entrada

Ejemplo:      valor = geo_InterpolaMallaOct(matrizVentana,posX,posY,ventana);

Funciones relacionadas de GEO: Ninguna

Ultima revision: 03-03-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PolLegendreOct.cc**:

```
Funcion:      geo_PolLegendreOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:    Calcula el valor de los polinomios de Legendre desde grado 0 a
              grado N
              La formulacion (formulas de recurrencia) ha sido extraida de
              [HM-GEFI-1985], pag. 23 y desarrollada a partir de
              [AS-UB91-1992], pag. 25

Entradas:     - Grado maximo del polinomio a calcular, numero entero >= 0
              - Argumento del polinomio (en el caso que nos ocupa, coseno de la
                colatitud esferica del punto de trabajo)
              - Cadena de texto 'UN' o 'FN' o 'QN', segun se quieran calcular
                los polinomios sin normalizar (UNNORMALIZED), fuertemente
                normalizados (FULLY-NORMALIZAED) en el sentido de
                [HM-GEFI-1985], pag. 23 o cuasi-normalizados (QUASI-NORMALIZED)
                en el sentido de [AS-UB91-1992], pag. 25

Salidas:      - Vector columna que contiene los valores del polinomio para el
                argumento pasado desde el grado 0 hasta el grado maximo
                indicado
                P = [ P0
                      P1
                     ....
                      PN];
                Cada fila corresponde a un polinomio distinto
                De esta forma, el indice de la columna en la matriz corresponde
                al grado del polinomio-1

Ejemplo:      [Pn]=geo_PolLegendreOct(grado,t,'UN');
              [Pn]=geo_PolLegendreOct(grado,t,'FN');
              [Pn]=geo_PolLegendreOct(grado,t,'QN');

Ultima revision: 16-10-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbDirGeoElipOct.cc**:

```
Funcion:    geo_ProbDirGeoElipOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve el problema directo de la geodesia sobre el elipsoide
            por el metodo de integracion numerica de Runge-Kutta de 4o orden
            La formulacion ha sido extraida de [GA-HL-GEO], pags. 211-215

Entradas:   - Latitud geodesica del punto inicial (escalar o vector columna),
              en radianes
            - Longitud geodesica del punto inicial (escalar o vector
              columna), en radianes
            - Acimut de la linea geodesica en el punto de partida (escalar o
              vector columna), en radianes
            - Longitud de la linea geodesica (escalar o vector columna)
            - Semieje mayor del elipsoide, en las mismas unidades que la
              longitud de la linea geodesica
            - Aplanamiento del elipsoide, (a-b)/a
            - Tolerancia lineal. Discrepancia minima entre dos distancias
              para que sean consideradas distintas en un proceso iterativo
            - Tolerancia angular. Discrepancia minima entre dos angulos para
              que sean considerados distintos
            Los argumentos que puedan ser escalares o vectores han de tener
            todos el mismo numero de filas

Salidas:    - Latitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Acimut reciproco de la linea geodesica (escalar o vector
              columna), en radianes

Ejemplo:    [lat2,lon2,aci21]=geo_ProbDirGeoElipOct(lat1,lon1,aci12,dist12,a,
            f,tolLineal,tolAngular);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_ProbInvGeoElipOct.oct

Ultima revision: 07-11-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbDirGeoEsfOct.cc**:

```
Funcion:    geo_ProbDirGeoEsfOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve el problema directo de la geodesia sobre la esfera por
            el metodo del cambio de base
            La formulacion ha sido extraida de [GA-HL-GEO], pags. 206-207

Entradas:   - Latitud geodesica del punto inicial (escalar o vector columna),
              en radianes
```

```
              - Longitud geodesica del punto inicial (escalar o vector
                columna), en radianes
              - Acimut de la linea geodesica en el punto de partida (escalar o
                vector columna), en radianes
              - Longitud de la linea geodesica (escalar o vector columna)
              - Radio de la esfera, en las mismas unidades que la longitud de
                la linea geodesica
              - Tolerancia. Discrepancia minima entre dos angulos para que sean
                considerados distintos, en radianes
              Los argumentos que puedan ser escalares o vectores han de tener
              todos el mismo numero de filas

   Salidas:   - Latitud geodesica del punto de llegada (escalar o vector
                columna), en radianes
              - Longitud geodesica del punto de llegada (escalar o vector
                columna), en radianes
              - Acimut reciproco de la linea geodesica (escalar o vector
                columna), en radianes

   Ejemplo:   [lat2,lon2,aci21]=geo_ProbDirGeoEsfOct(lat1,lon1,aci12,dist12,
              radio,tol);

   Funciones utilizadas de GEO: Ninguna

   Funciones relacionadas de GEO: - geo_ProbInvGeoEsfOct.oct

   Ultima revision: 02-10-06

   Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbInvGeoElipOct.cc**:

```
   Funcion:   geo_ProbInvGeoElipOct.cc

   Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

   Proposito: Resuelve el problema inverso de la geodesia sobre el elipsoide
              por el metodo de Bessel modificado
              La formulacion ha sido extraida de [GA-HL-GEO], pags. 215-232

   Entradas:  - Latitud geodesica del punto de partida (escalar o vector
                columna), en radianes
              - Longitud geodesica del punto de partida (escalar o vector
                columna), en radianes
              - Latitud geodesica del punto de llegada (escalar o vector
                columna), en radianes
              - Longitud geodesica del punto de llegada (escalar o vector
                columna), en radianes
              - Semieje mayor del elipsoide
              - Aplanamiento del elipsoide, (a-b)/a
              - Tolerancia angular. Discrepancia minima entre dos angulos para
                que sean considerados distintos en un proceso iterativo
              Los argumentos que puedan ser escalares o vectores han de tener
              todos el mismo numero de filas

   Salidas:   - Longitud de la linea geodesica (escalar o vector columna), en
                las mismas unidades que el semieje mayor del elipsoide
              - Acimut directo (del punto de partida al de llegada) de la linea
```

```
              geodesica (escalar o vector columna), en radianes
            - Acimut reciproco (del punto de llegada al de partida) de la
              linea geodesica (escalar o vector columna), en radianes

Ejemplo:    [dist12,aci12,aci21]=geo_ProbInvGeoElipOct(lat1,lon1,lat2,lon2,a,
            f,tolAngular);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_ProbDirGeoElipOct.oct

Ultima revision: 11-03-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbInvGeoEsfOct.cc**:

```
Funcion:    geo_ProbInvGeoEsfOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve el problema inverso de la geodesia sobre la esfera por
            el metodo del cambio de base
            La formulacion ha sido extraida de [GA-HL-GEO], pags. 204-205

Entradas:   - Latitud geodesica del punto de partida (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de partida (escalar o vector
              columna), en radianes
            - Latitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Radio de la esfera
            - Tolerancia. Discrepancia minima entre dos angulos para que sean
              considerados distintos, en radianes
            Los argumentos que puedan ser escalares o vectores han de tener
            todos el mismo numero de filas

Salidas:    - Longitud de la linea geodesica, en las mismas unidades que el
              radio de la esfera
            - Acimut directo (del punto de partida al de llegada) de la linea
              geodesica (escalar o vector columna), en radianes
            - Acimut reciproco (del punto de llegada al de partida) de la
              linea geodesica (escalar o vector columna), en radianes

Ejemplo:    [dist12,aci12,aci21]=geo_ProbInvGeoEsfOct(lat1,lon1,lat2,lon2,
            radio,tol);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_ProbDirGeoEsfOct.oct

Ultima revision: 02-10-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ResTriInfOct.cc**:

```
Funcion:    geo_ResTriInfOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve un sistema de ecuaciones, Ax=B, con la matriz A
            triangular inferior

Entradas:   - Matriz A, cuadrada y triangular inferior
            - Vector de terminos independientes B. Dos posibilidades:
              - Vector columna, con el mismo numero de filas que la matriz A
              - Matriz con el mismo numero de filas que la matriz A, donde
                cada columna sera considerada como un vector B distinto

Salidas:    - Incognitas del sistema. Dos posibilidades:
              - Vector columna con la solucion del sistema en el caso que el
                termino B del sistema haya sido un vector
              - Matriz en la que cada columna es un vector solucion del
                en el caso que el termino B del sistema haya sido una matriz

Ejemplo:    [x]=geo_ResTriInfOct(A,B);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_ResTriSupOct.oct

Ultima revision: 01-10-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ResTriSupOct.cc**:

```
Funcion:    geo_ResTriSupOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve un sistema de ecuaciones, Ax=B, con la matriz A
            triangular superior

Entradas:   - Matriz A, cuadrada y triangular superior
            - Vector de terminos independientes B. Dos posibilidades:
              - Vector columna, con el mismo numero de filas que la matriz A
              - Matriz con el mismo numero de filas que la matriz A, donde
                cada columna sera considerada como un vector B distinto

Salidas:    - Incognitas del sistema. Dos posibilidades:
              - Vector columna con la solucion del sistema en el caso que el
                termino B del sistema haya sido un vector
              - Matriz en la que cada columna es un vector solucion del
                en el caso que el termino B del sistema haya sido una matriz

Ejemplo:    [x]=geo_ResTriSupOct(A,B);

Funciones utilizadas de GEO: Ninguna

Funciones relacionadas de GEO: - geo_ResTriInfOct.oct

Ultima revision: 01-10-06
```

```
Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_TestEndianOct.cc**:

```
Funcion:    geo_TestEndianOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Determina el tipo de almacenamiento binario de la maquina

Entradas:   Ninguna

Salidas:    - Cadena de texto 'LITTLE-ENDIAN' o 'BIG-ENDIAN', dependiento del
              tipo de almacenamiento

Ejemplo:    [tipoAlmacenamiento]=geo_TestEndianOct;

Ultima revision: 04-02-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_TriangleOct.cc**:

```
Funcion:    geo_TriangleOct.cc

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Genera un triangulacion de Delaunay utilizando la biblioteca
            Triangle

Entradas:   - Matriz de Nx2 elementos que almacena las coordenadas en el
              plano de los puntos de la nube a triangular (ha de haber un
              minimo de 4 puntos). Cada fila corresponde a un punto:
              - Col. 1: Coordenada X
              - Col. 2: Coordenada Y
            - Matriz vacia o de Lx2 elementos que almacena los segmentos que
              forman el borde de la nube de puntos a triangular. Estos
              segmentos tambien funcionan como constrenimientos, cada uno de
              los cuales solo formara parte de un triangulo. Cada fila de
              esta matriz corresponde a un segmento:
              - Col. 1: Posicion en la matriz de puntos pasada del nodo
                        inicial de la linea
              - Col. 2: Posicion en la matriz de puntos pasada del nodo final
                        de la linea
              Si este argumento es una matriz vacia el programa de
              triangulacion generara un borde para la nube de puntos
              Todos los puntos de la nube y lineas de rotura que queden
              fuera de los limites definidos no seran tenidos en cuenta
              El limite definido ha de ser un poligono cerrado, ha de partir
              y llegar al mismo punto de la nube o no se generara
              triangulacion alguna
            - Matriz vacia o de Rx2 elementos que almacena las lineas de
              rotura o segmentos de constrenimiento a aplicar a la
              triangulacion. De esta forma se obliga a que los segmentos
              indicados sean lados de algun triangulo de la triangulacion
              Imponer estas lines implica que algunos triangulos pueden no
              respetar la condicion de Delaunay. Cada fila de esta matriz
```

```
                corresponde a un segmento:
                - Col. 1: Posicion en la matriz de puntos pasada del nodo
                          inicial de la linea
                - Col. 2: Posicion en la matriz de puntos pasada del nodo final
                          de la linea
                Si este argumento es una matriz vacia no se tendran en cuenta
                lineas de rotura
                Hay que evitar que dos o mas lineas de rotura se corten ya
                que dara lugar a un funcionamiento incorrecto del programa de
                triangulacion
                Hay que evitar que el borde y alguna linea de rotura se corten
                ya que dara lugar a un funcionamiento incorrecto del programa
                de triangulacion
              - H Matriz vacia o de Hx2 elementos que almacena los posible
                huecos que haya en la nube de puntos y que no es necesario
                triangular. Cada fila de esta matriz corresponde a un punto que
                esta dentro del hueco definido por lineas de rotura en la
                matriz correspondiente:
                - Col. 1: Coordenada X de un punto que esta dentro de un hueco
                - Col. 2: Coordenada Y de un punto que esta dentro de un hueco
                Si se introducen huecos cuyo punto identificador no esta dentro
                de un poligono definido por lineas de rotura no se creara
                ningun triangulo en la nube de puntos
              - Cadena de texto con las opciones a pasar al programa de
                triangulacion. Cada caracter en la cadena sera una opcion de
                entre las siguientes:
                - 'i': Se utilizara el algoritmo incremental para generar la
                       triangulacion en lugar del de divide y venceras (divide
                       and conquer)
                - 'F': Se utilizara el algoritmo sweepline de Steven Fortune
                       para generar la triangulacion en lugar del de divide y
                       venceras (divide and conquer)
                - 'Q': Elimina la informacion de procesado que aparece en
                       pantalla mientras se crea la malla. Los mensajes de
                       error no los elimina
                En el caso de seleccionar a la vez las opciones 'i' y 'F' (sea
                en el orden que sea) se utilizara el algoritmo incremental
                Para mas informacion acerca de las opciones ver la pagina web
                de ayuda de la biblioteca Triangle (ver Nota al final de la
                ayuda)

Salidas:      - Matriz de Tx3 elementos que contiene la triangulacion de
                Delaunay de la nube de puntos pasada. Cada fila describe cada
                uno de los T triangulos que la forman. Cada columna es la
                posicion en la matriz de la nube de puntos que corresponde a
                cada vertice del triangulo

Ejemplo:      T=geo_TriangleOct(puntos,limites,rotura,huecos,opciones);

Funciones relacionadas de GEO: Ninguna

Nota: La biblioteca Triangle ha sido creada por Jonathan Richard Shewchuk e
      implementa un algoritmo mas rapido que el de Qhull (que utilizan
      GNU/Octave y MatLab) para generar triangulaciones de Delaunay. Tambien
      es posible aplicar constrenimientos a la triangulacion. Esta biblioteca
      no se distribuye bajo la licencia GPL, sino que posee una propia. Para
      mas informacion visitar la web:
```

```
      http://www.cs.cmu.edu/~quake/triangle.html

Ultima revision: 04-09-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

### 10.14.2. Carpeta 02-oct

En esta carpeta se almacenan las funciones de GEO escritas en C++ ya compiladas.

### 10.14.3. Carpeta 03-compilacion

En esta carpeta se guarda el *script* de compilación de las funciones de GEO escritas en C++.

- **compilaOct.sh**:

```
Funcion:    compilaOct.sh

Proposito:  Script para shell (bash) que compila las funciones ecritas en
            C/C++ y/o Fortran para GEO

Entradas:   - Ruta del directorio de los archivos de cabecera (*.h) de GEO
            - Ruta del directorio de los archivos fuente (*.c, *.cc) de las
              librerias de GEO
            - Ruta del directorio de los archivos fuente (*.cc) de las
              funciones de GEO
            - Ruta del directorio de almacenamiento de las funciones de
              GNU Octave (*.oct) generadas
            - Indicador de borrado de archivos intermedios 1/0 -> si/no

Salidas:    - Ficheros en C++ compilados (.oct) y listos para ser usados. Se
              guardan en la carpeta correspondiente al cuarto argumento de
              entrada. En el caso de no borrar los archivos intermedios, estos
              se almacenaran en el mismo directorio que los ejecutables
              creados

Ejemplo:    sh compilaOct.sh rutaInclude rutaLibsCpp rutaOctCpp rutaOct borrar

Usa geoVarGlobal: Si

Nota:       Este script solo funciona en sistemas GNU/Linux-UNIX. No lo
            modifique en sistemas Windows ya que al guardar los cambios,
            debido a la diferente codificacion de los sistemas, se modificaran
            los indicadores de fin de linea y dara errores en bash. Si lo
            modifica, puede cambiar la codificacion interna con el programa
            dos2unix en sistemas GNU/Linux-UNIX

Original:   22-08-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 18-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
```

## 10.15. Carpeta 15-generadorAyuda

En esta carpeta se almacenan las funciones relacionadas con la generación automática de ayuda.

- **geo_ExtraeAyudaLaTeX.m**:

```
Funcion:    geo_ExtraeAyudaLaTeX.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Extrae la ayuda de una funcion (*.m), un fichero de datos (*.dat),
            un fichero C++ (*.cc o *.h) o un script de Bash (*.sh) de GEO a un
            fichero de LaTeX en el entorno \verbatim

Entradas:   - Direccion del fichero a leer, como cadena de texto
            - Nombre del fichero a leer, como cadena de texto

Salidas:    - Crea un fichero de LaTeX en la carpeta de trabajo con la ayuda
              contenida en el fichero: fichero.m.tex, fichero.dat.tex, etc

Ejemplo:    geo_ExtraeAyudaLaTeX('direccion/funcion.m','funcion.m');
            geo_ExtraeAyudaLaTeX('direccion/datos.dat','datos.dat');

Funciones utilizadas de GEO: - geo_EsTexto.m

Funciones relacionadas de GEO: - geo_GeneraAyudaFuncionesLaTeX.m
                               - geo_GeneraAyudaLaTeX.m
                               - geo_GeneraAyudaUsuarioLaTeX.m
                               - geo_GeneraPorHacerLaTeX.m
                               - geo_PortadaAyudaLaTeX.m

Ultima revision: 19-08-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GeneraAyudaFuncionesLaTeX.m**:

```
Funcion:    geo_GeneraAyudaFuncionesLaTeX.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Escribe en un archivo LaTeX la ayuda de todas las funciones y
            ficheros de datos de GEO

Entradas:   - Dos posibilidades:
              - Nombres de los autores, como cadenas de texto. En este caso se
                genera un archivo listo para compilar con la ayuda
              - Ningun argumento. En este caso se escribe un documento de
                LaTeX para ser incluido otro documento compilable

Salidas:    - Con argumentos de entrada:
              - Crea la carpeta 'ayudaFuncionesGeo' (en el directorio de
                trabajo), que contiene todos los ficheros necesarios para
                generar la ayuda
            - Sin argumentos de entrada
              - Crea la carpeta 'ayudaFunciones' (en el directorio de
                trabajo), que contiene todos los ficheros necesarios para
                incluir la ayuda del usuario y del desarrollador en un fichero
                compilable de LaTeX

Ejemplo:    geo_GeneraAyudaFuncionesLaTeX(nombre1,nombre2,...);
            geo_GeneraAyudaFuncionesLaTeX;

Funciones utilizadas de GEO: - geo_EsTexto.m
```

```
                                - geo_DesactivaControles.m
                                - geo_CopiaArchivo.m
                                - geo_PortadaAyudaLaTeX.m
                                - geo_LeeDirectorio.m
                                - geo_RestauraControles.m
                                - geo_ExtraeAyudaLaTeX.m

Funciones relacionadas de GEO: - geo_ExtraeAyudaLaTeX.m
                                - geo_GeneraAyudaLaTeX.m
                                - geo_GeneraAyudaUsuarioLaTeX.m
                                - geo_GeneraPorHacerLaTeX.m
                                - geo_PortadaAyudaLaTeX.m

Ultima revision: 02-05-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GeneraAyudaLaTeX.m**:

```
Funcion:    geo_GeneraAyudaLaTeX.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Escribe en un archivo LaTeX la ayuda de GEO

Entradas:   - Nombres de los autores, como cadenas de texto

Salidas:    - Crea la carpeta 'ayudaGeo' (en el directorio de trabajo), que
              contiene todos los ficheros necesarios para generar con LaTeX el
              manual de referencia de GEO

Ejemplo:    geo_GeneraAyudaLaTeX(nombre1,nombre2,...);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_DesactivaControles.m
                             - geo_CopiaArchivo.m
                             - geo_PortadaAyudaLaTeX.m
                             - geo_GeneraAyudaUsuarioLaTeX.m
                             - geo_GeneraAyudaFuncionesLaTeX.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_ExtraeAyudaLaTeX.m
                               - geo_GeneraAyudaFuncionesLaTeX.m
                               - geo_GeneraAyudaUsuarioLaTeX.m
                               - geo_GeneraPorHacerLaTeX.m
                               - geo_PortadaAyudaLaTeX.m

Ultima revision: 02-05-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GeneraAyudaUsuarioLaTeX.m**:

```
Funcion:    geo_GeneraAyudaUsuarioLaTeX.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
Proposito:  Escribe en un archivo LaTeX las ayudas del usuario y del
            desarrollador de GEO

Entradas:   - Dos posibilidades:
              - Nombres de los autores, como cadenas de texto. En este caso se
                genera un archivo listo para compilar con la ayuda
              - Ningun argumento. En este caso se escribe un documento de
                LaTeX para ser incluido otro documento compilable

Salidas:    - Con argumentos de entrada:
              - Crea la carpeta 'ayudaUsuarioGeo' (en el directorio de
                trabajo), que contiene todos los ficheros necesarios para
                generar la ayuda
            - Sin argumentos de entrada
              - Crea las carpetas 'ayudaUsuario' y 'ayudaDesarrollador' (en el
                directorio de trabajo), que contienen todos los ficheros
                necesarios para incluir la ayuda del usuario y del
                desarrollador en un fichero compilable de LaTeX

Ejemplo:    geo_GeneraAyudaUsuarioLaTeX(nombre1,nombre2,...);
            geo_GeneraAyudaUsuarioLaTeX;

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_DesactivaControles.m
                             - geo_CopiaArchivo.m
                             - geo_PortadaAyudaLaTeX.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_ExtraeAyudaLaTeX.m
                               - geo_GeneraAyudaFuncionesLaTeX.m
                               - geo_GeneraAyudaLaTeX.m
                               - geo_GeneraPorHacerLaTeX.m
                               - geo_PortadaAyudaLaTeX.m

Ultima revision: 02-05-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GeneraPorHacerLaTeX.m**:

```
Funcion:    geo_GeneraPorHacerLaTeX.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Escribe en un archivo LaTeX las tareas pendientes en la
            programacion de GEO

Entradas:   Ninguna

Salidas:    - Crea la carpeta 'porHacerGeo' (en el directorio de trabajo), que
              contiene todos los ficheros necesarios para generar con LaTeX un
              documento con las tareas pendientes de GEO

Ejemplo:    geo_GeneraPorHacerLaTeX;

Funciones utilizadas de GEO: - geo_DesactivaControles.m
                             - geo_CopiaArchivo.m
                             - geo_RestauraControles.m
```

```
Funciones relacionadas de GEO: - geo_ExtraeAyudaLaTeX.m
                               - geo_GeneraAyudaFuncionesLaTeX.m
                               - geo_GeneraAyudaLaTeX.m
                               - geo_GeneraAyudaUsuarioLaTeX.m
                               - geo_PortadaAyudaLaTeX.m

Ultima revision: 25-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PortadaAyudaLaTeX.m**:

```
Funcion:    geo_PortadaAyudaLaTeX.m

Proposito:  Genera un archivo, en formato LaTeX, con la portada para el manual
            de ayuda correspondiente

Entradas:   - Titulo del documento, como cadena de texto
            - Subtitulo del documento, como cadena de texto
            - Cell array que contiene el nombre de los autores como cadenas de
              texto. No se controla internamente si los elementos del cell
              array son cadenas de texto

Salidas:    - Genera un fichero *.tex en el directorio de trabajo llamado
              'portadaAyudaGeo.tex', que contiene el texto para la portada del
              documento de ayuda. El documento termina con un salto de pagina
              (\newpage), por lo que no es necesario incluir este comando en
              el fichero de ayuda
            - Si todo ha ido bien, la funcion devuelve el nombre del fichero
              creado, como cadena de texto

Ejemplo:    geo_PortadaAyudaLaTeX(titulo,subtitulo,nombre);

Funciones utilizadas de GEO: - geo_EsTexto.m

Funciones relacionadas de GEO: - geo_ExtraeAyudaLaTeX.m
                               - geo_GeneraAyudaFuncionesLaTeX.m
                               - geo_GeneraAyudaLaTeX.m
                               - geo_GeneraAyudaUsuarioLaTeX.m
                               - geo_GeneraPorHacerLaTeX.m

Ultima revision: 02-05-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 10.16. Carpeta 16-informacion

En esta carpeta se almacenan las funciones relacionadas con la información al usuario sobre GEO.

- **geo_Bibliografia.m**:

```
Funcion:    geo_Bibliografia.m

Proposito:  Muestra informacion acerca de la bibliografia consultada en la
            programacion de GEO
```

```
Entradas:   - Dos argumentos de entrada:
              - Primer argumento (cuatro posibilidades):
                - Identificador de referencia bibliografica. Cadena de texto
                - Cadena de texto 'ARTICULOS'
                - Cadena de texto 'LIBROS'
                - Cadena de texto 'TODO'
              - Segundo argumento:
                - Cadena de tetxto que almacena el nombre del fichero que se
                  creara para almacenar las referencias extraidas
            - Un argumento de entrada:
              - Cuatro posibilidades:
                - Identificador de referencia bibliografica. Cadena de texto
                - Cadena de texto 'ARTICULOS'
                - Cadena de texto 'LIBROS'
                - Cadena de texto 'TODO'
            - Ningun argumento de entrada:
              - Es equivalente a pasar un solo argumento con la cadena 'TODO'


Salidas:    - Numero de referencias mostradas. Si no se muestra ninguna, este
              argumento vale 0
            - Impresion en pantalla o en fichero (se crea en el directorio de
              trabajo) de las referencias bibliograficas, dependiendo del
              primer argumento de entrada:
              - Si se ha pasado un identificador de referencia bibliografica
                se muestra la referencia correspondiente, si existe
              - Si se ha pasado la cadena 'ARTICULOS' se muestran todos los
                articulos de la bibliografia, ordenados alfabeticamente por
                identificador
              - Si se ha pasado la cadena 'LIBROS' se muestran todos los
                libros de la bibliografia, ordenados alfabeticamente por
                identificador
              - Si no se ha pasado ningun argumento de entrada o la cadena
                'TODOS' se muestran todos los elementos de la bibliografia,
                ordenados alfabeticamente por identificador

Ejemplo:    numero=geo_Bibliografia;
            numero=geo_Bibliografia('TODO');
            numero=geo_Bibliografia('LIBROS');
            numero=geo_Bibliografia('ARTICULOS');
            numero=geo_Bibliografia('ARTICULOS','ficheroArticulos.txt');

Usa geoVarGlobal: Si

Nota:       En las posibles cadenas de texto de entrada no se distingue entre
            mayusculas y minusculas

Original:   24-03-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reescritura completa de la funcion para que utilice el fichero
```

```
                de texto de bibliografia
              06-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Anyadida sentencia que lanza un warning si el fichero esta vacio
```

- **geo_Copyright.m**:

```
Funcion:    geo_Copyright.m

Proposito:  Muestra el mensaje de copyright de GEO. El mensaje mostrado es el
            que se considera oficial para GEO, por encima del indicado en la
            cabecera de todos los archivos de funcion

Entradas:   Ninguna

Salidas:    Mensaje de copyright en pantalla

Ejemplo:    geo_Copyright;

Usa geoVarGlobal: No

Original:   27-12-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 01-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

## 10.17. Carpeta 17-otras

En esta carpeta se almacenan las funciones que no encajan en ninguna de las demás categorías de la sección general.

- **geo_AcimutAnguloMate.m**:

```
Funcion:    geo_AcimutAnguloMate.m

Proposito:  Reduce un valor de acimut al dominio angular matematico ]-pi pi],
            con sentido antihorario y origen en la parte positiva del eje X

Entradas:   - Valor angular de acimut (escalar o vector columna), en radianes

Salidas:    - Variable reducida al dominio ]-pi pi] en sentido antihorario y
              con origen en la parte positiva del eje X (escalar o vector
              columna), en radianes

Ejemplo:    angulo=geo_AcimutAnguloMate(acimut);

Usa geoVarGlobal: Si

Original:   12-08-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_ActivaControles.m**:

```
Funcion:    geo_ActivaControles.m

Proposito:  Activa los controles indicados como argumentos de entrada

Entradas:   - Ningun argumento o una serie de cadenas de texto con el nombre
              del tipo de control a activar. Las cadenas de texto permitidas
              son (en mayusculas o minusculas):
              - CONDICIONES: Activa el control de condiciones especiales
              - DIMENSIONES: Activa el control de dimensiones de los datos
              - DOMINIO: Activa el control de dominio de dato
              - TIPODATO: Activa el control de tipo de dato
              - TIEMPOFICHERO: Activa el control de tiempo en fichero
              - TIEMPOPANTALLA: Activa el control de tiempo en pantalla

Salidas:    - Activa los controles pedidos en la variable geoVarGlobal
            - Estructura con el estado previo de los controles (solo existen
              los campos correspondientes a los controles que han sido
              activados):
              - previo.CONDICIONES
              - previo.DIMENSIONES
              - previo.DOMINIO
              - previo.TIPODATO
              - previo.TIEMPOFICHERO
              - previo.TIEMPOPANTALLA
              Hay un campo que existe siempre y que indica si se ha activado
              algun control (0/1 -> no/si):
              - previo.CAMBIOS

Ejemplo:    previo=geo_ActivaControles;
            previo=geo_ActivaControles('CONDICIONES','DOMINIO',...);

Usa geoVarGlobal: Si

Nota:       Esta funcion no controla internamente el tipo de dato pasado

Original:   25-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geo_AjustaFilasVectorMatriz.m**:

```
Funcion:    geo_AjustaFilasVectorMatriz.m

Proposito:  Convierte un vector fila en una matriz cuyas filas son el vector
            fila original

Entradas:   - Vector fila o matriz original
            - Filas de la matriz final
            Argumentos opcionales:
              - Nombre de la matriz original, como cadena de texto
              - Nombre de la matriz final, como cadena de texto

Salidas:    - Una matriz con un numero de filas igual al pasado con el vector
              original en cada una de sus filas. Si el primer argumento es una
              matriz cuyas filas no coinciden con las que ha de tener la
              matriz de salida se lanza un error

Ejemplo:    matriz=geo_AjustaFilasVectorMatriz(vector,filas);
```

```
            matriz=geo_AjustaFilasVectorMatriz(vector,filas,nombreVector,
            nombreMatriz);

  Usa geoVarGlobal: No

  Nota:       Esta funcion no controla internamente el tipo de los argumentos
              pasados

  Original:   08-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: 29-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Reorganizacion de la ayuda
                Reescritura de la funcion para que no utilice la variable
                varargin como contenedor de los argumentos de entrada
                Adicion de sentencias de test
```

- **geo_AjustaTexto.m**:

```
  Funcion:    geo_AjustaTexto.m

  Proposito:  Ajusta y justifica una cadena de texto a la longitud indicada,
              sin tener en cuenta los posibles espacios en blanco al principio y
              final en la cadena original

  Entradas:   - Una cadena de texto
              - Longitud final de la cadena
              Argumentos opcionales:
                - Tipo de justificacion, cadenas: 'IZQUIERDA', 'DERECHA' o
                  'CENTRO'. Por omision, este argumento vale 'IZQUIERDA'

  Salidas:    - La cadena ajustada

  Ejemplo:    cadena=geo_AjustaTexto('ACIMUT',20,);
              cadena=geo_AjustaTexto('ACIMUT',20,'DERECHA');

  Usa geoVarGlobal: Si

  Nota:       En el caso de que haya que centrar el texto y sobre un espacio,
              este se almacena a la derecha de la cadena de salida

  Original:   25-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: 29-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Reorganizacion de la ayuda
                Reescritura de la funcion para que no utilice la variable
                varargin como contenedor de los argumentos de entrada
                Adicion de sentencias de test
```

- **geo_AnguloMateAcimut.m**:

```
  Funcion:    geo_AnguloMateAcimut.m

  Proposito:  Reduce un valor del dominio angular matematico ]-pi pi], con
              sentido antihorario y origen en la parte positiva del eje X, al
              dominio angular del acimut

  Entradas:   - Valor angular en dominio matematico (escalar o vector columna),
```

```
              en radianes

Salidas:    - Variable reducida al dominio del acimut, en radianes

Ejemplo:    acimut=geo_AnguloMateAcimut(angulo);

Usa geoVarGlobal: Si

Original:   12-08-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_BuscaNumero.m**:

```
Funcion:    geo_BuscaNumero.m

Proposito:  Busca un numero (la primera ocurrencia) en un vector (fila o
            columna) o en una fila o columna determinada de una matriz

Entradas:   - Numero a buscar, que puede ser un escalar o un vector columna
            - La variable donde se realizara la busqueda
              Si la variable donde vamos a buscar es una matriz:
              - Cadena de texto 'FILA' o 'COLUMNA', que indica la dimension
                donde vamos a buscar
              - Numero de la fila o columna donde vamos a buscar

Salidas:    - Una matriz de N (cantidad de numeros a buscar) filas por tres
              columnas con los resultados de la busqueda. Las dos primeras
              columnas de cada fila corresponden a las coordenadas
              (fila-columna) de la primera ocurrencia del numero buscado en la
              matriz de busqueda. La tercera columna de cada fila almacena el
              numero de ocurrencias del numero. Si el dato no se ha
              encontrado, la posicion devuelta es [0 0 0]

Ejemplo:    posiciones=geo_BuscaNumero(numeros,vector);
            posiciones=geo_BuscaNumero(numeros,matriz,'COLUMNA',numeroFilCol);

Usa geoVarGlobal: Si

Original:   25-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 02-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_ComentariosFichero.m**:

```
Funcion:    geo_ComentariosFichero.m

Proposito:  Determina el numero de lineas de comentario existentes en un
            fichero
```

```
Entradas:   - Ruta del fichero de trabajo, como cadena de texto

Salidas:    - Numero de lineas de comentario en el fichero. Numero entero

Ejemplo:    nLineas=geo_ComentariosFichero('nombreFichero');

Usa geoVarGlobal: Si

Original:   25-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 17-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            18-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de los argumentos de entrada
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la funcion. Ahora solo cuenta el numero de
              lineas de comentario en el fichero. Para el posicionamiento del
              puntero de L/E en la primera linea no comentada o no blanca
              utilizar la funcion geo_PosFicheroLineaDatos.m
```

- **geo_CompilaOct.m**:

```
Funcion:    geo_CompilaOct.m

Proposito:  Compila todas las funciones de GEO escritas en C/C++ y/o Fortran

Entradas:   Tres posibilidades:
            - Ningun argumento de entrada:
               Guarda las funciones *.oct creadas en la carpeta definida en la
               variable global geoVarGlobal.carpeta.binOct y borra los
               archivos intermedios producidos durante la compilacion
            - Un argumento de entrada:
              Dos posibilidades:
              - Ruta del directorio donde se guardaran las funciones
                compiladas, como cadena de texto. En este caso se borraran los
                ficheros auxiliares
              - Indicador de borrado de ficheros auxiliares 1/0 -> si/no. En
                este caso las funciones compiladas se guardaran en la carpeta
                definida en la varable global geoVarGlobal.carpeta.binOct
            - Dos argumentos de entrada:
              - Ruta del directorio donde se guardaran las funciones
                compiladas, como cadena de texto
              - Indicador de borrado de ficheros auxiliares 1/0 -> si/no

Salidas:    - Funciones compiladas en la carpeta indicada y ficheros
              intermedios, si ha lugar
            - Archivo 'geo_CompilaOct.log', en el directorio actual, con los
              resultados de la compilacion

Ejemplo:    geo_CompilaOct;
            geo_CompilaOct(directorioOct);
            geo_CompilaOct(indicadorBorrado);
            geo_CompilaOct(directorioOct,indicadorBorrado);
```

```
Usa geoVarGlobal: Si

Nota:       Esta funcion solo es operativa en GNU Octave. El directorio donde
            se guardaran las funciones compiladas ha de tener permiso de
            escritura

Original:   02-07-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 18-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_CompruebaColumnas.m**:

```
Funcion:    geo_CompruebaColumnas.m

Proposito:  Comprueba si los elementos pasados tienen un numero correcto de
            columnas

Entradas:   Grupos de tres argumentos de la siguiente manera:
              - Numero de columnas de la matriz a comprobar, numero natural
              - Vector fila con el numero de posibles valores de columnas que
                la matriz a comprobar puede tener
              - Nombre de la matriz a comprobar, como cadena de texto

Salidas:    - 1, si el numero de columnas de las matrices es correcto
            - Un error en caso contrario

Ejemplo:    geo_CompruebaColumnas(colMat1,[col1 col2 col3],'mat1',...);

Usa geoVarGlobal: No

Nota:       Esta funcion no comprueba internamente el tipo de sus argumentos
            de entrada

Original:   08-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 18-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

- **geo_Concatena.m**:

```
Funcion:    geo_Concatena.m

Proposito:  Concatena en una sola varias cadenas de texto

Entradas:   - Una cadena de texto, a la cual se uniran las demas
            - Otra cadena de texto
            - ...

Salidas:    - Las cadenas de entrada fusionadas en una sola
            Si solo se trabaja con un argumento de entrada, la cadena devuelta
            es identica a la pasada
```

```
Ejemplo:    cadena=geo_Concatena('acimut',' ','geodesico');

Usa geoVarGlobal: Si

Original:   11-03-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 18-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

- **geo_CopiaArchivo.m**:

```
Funcion:    geo_CopiaArchivo.m

Proposito:  Copia un archivo de una ubicacion dada a una nueva direccion

Entradas:   - Ruta del archivo a copiar, como cadena de texto. Puede ser
              relativa al directorio donde nos encontremos o absoluta
            - Ruta destino, como cadena de texto. Puede ser relativa al
              directorio donde nos encontremos o absoluta

Salidas:    - 1, si la operacion se ha realizado correctamente
            - Un error, en caso contrario

Ejemplo:    estado=geo_CopiaArchivo('ruta1/archivo','ruta2/');

Usa geoVarGlobal: Si

Nota:       Esta funcion tiene el mismo comportamiento que la funcion
            'copyfile' de GNU Octave o MatLab. En algunas versiones antiguas
            de GNU Octave no existe la funcion 'copyfile', por lo que para
            llevar a cabo la tarea se utilizaran las funciones 'cp' o 'copy'
            del sistema, dependiendo de si entamos trabajando en UNIX/Linux o
            Windows, respectivamente

Original:   23-05-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 18-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_CreaLineaPuntosElipsoide.m**:

```
Funcion:    geo_CreaLineaPuntosElipsoide.m

Proposito:  Crea una sucesion de puntos coordenados equiespaciados sobre el
            elipsoide

Entradas:   - Numero de puntos a crear (incluidos los extremos), numero
              natural
            - Latitud geodesica del punto de inicio, en radianes
            - Longitud geodesica del punto de inicio, en radianes
            - Latitud geodesica del punto de llegada, en radianes
            - Longitud geodesica del punto de llegada, en radianes
              Para los parametros de entrada, el dominio de la latitud es
```

```
              [-pi/2 pi/2] y el de la longitud ]-pi pi]
            Argumentos opcionales:
              - Altitud elipsoidal del punto de inicio
              - Altitud elipsoidal del punto de llegada

Salidas:    - Matriz con la siguiente estructura:
              - Col. 1: Latitud geodesica de cada punto
              - Col. 2: Longitud geodesica de cada punto
              Si se ha indicado altitud elipsoidal:
                - Col. 3: Altitud elipsoidal de cada punto

Ejemplo:    puntos=geo_CreaLineaPuntosElipsoide(nPuntos,latIni,lonIni,latFIn,
            lonFin);
            puntos=geo_CreaLineaPuntosElipsoide(nPuntos,latIni,lonIni,latFIn,
            lonFin,hIni,hFin);

Usa geoVarGlobal: Si

Nota:       El sentido de la linea creda en longitud sera siempre de oeste a
            este, entendiendo como oeste la longitud de inicio y como este la
            de fin. Una linea de 5 puntos que vaya desde la longitud 170 a la
            -170 pasara por los puntos [170 175 180 -175 -170] mientras que
            una que vaya desde la longitud -170 a 170 pasara por los puntos
            [-170 -85 0 85 170]

Original:   24-02-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 19-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_CreaLineaPuntosPlano.m**:

```
Funcion:    geo_CreaLineaPuntosPlano.m

Proposito:  Crea una sucesion de puntos coordenados equiespaciados sobre el
            plano

Entradas:   - Numero de puntos a crear (incluidos los extremos), numero
              natural
            - Coordenada X del punto de inicio
            - Coordenada Y del punto de inicio
            - Coordenada X del punto de llegada
            - Coordenada Y del punto de llegada
            Argumentos opcionales:
              - Coordenada Z del punto de inicio
              - Coordenada Z del punto de llegada

Salidas:    - Matriz con la siguiente estructura:
              - Col. 1: Coordenada X de cada punto
              - Col. 2: Coordenada Y de cada punto
              Si se ha indicado coordenada Z:
                - Col. 3: Coordenada Z de cada punto

Ejemplo:    puntos=geo_CreaLineaPuntosPlano(100,Xi,Yi,Xf,Yf);
            puntos=geo_CreaLineaPuntosPlano(100,Xi,Yi,Xf,Yf,Zi,Zf);
```

```
Usa geoVarGlobal: Si

Original:   24-02-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 19-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_CreaMallaPuntosElipsoide.m**:

```
Funcion:    geo_CreaMallaPuntosElipsoide.m

Proposito:  Crea una malla de puntos coordenados sobre el elipsoide

Entradas:   - Latitud geodesica del punto de la esquina noroeste, en radianes
            - Longitud geodesica del punto de la esquina noroeste, en radianes
            - Latitud geodesica del punto de la esquina sureste, en radianes
            - Longitud geodesica del punto de la esquina sureste, en radianes
            - Paso de malla para la latitud geodesica (valor absoluto),
              en radianes
            - Paso de malla para la longitud geodesica (valor absoluto),
              en radianes
              Para los parametros de entrada, el dominio de la latitud es
              [-pi/2 pi/2] y el de la longitud ]-pi pi]
            Argumentos opcionales:
              - Altitud elipsoidal del punto de la esquina noroeste
              - Altitud elipsoidal del punto de la esquina sureste

Salidas:    - Matriz con la siguiente estructura:
              - Col. 1: Latitud geodesica de cada punto
              - Col. 2: Longitud geodesica de cada punto
              Si se ha indicado altitud elipsoidal:
                - Col. 3: Altitud elipsoidal de cada punto. El paso en altitud
                          se calcula a partir del numero de puntos y de la
                          diferencia entre la altitud de la esquina sureste
                          menos la de la esquina noroeste
            - Numero de filas de la malla creada (puntos en latitud)
            - Numero de columnas de la malla creada (puntos en longitud)
              Los puntos de la malla se representan de oeste a este y de norte
              a sur. Es decir, la malla:
                                              [lat1 lon1 |h1|
              1  2                             lat2 lon2 |h2|
              3  4  queda representada como    lat3 lon3 |h3|
              5  6                             lat4 lon4 |h4|
                                               lat5 lon5 |h5|
                                               lat6 lon6 |h6|]
              Si se ha solicitado calculo de altura elipsoidal, esta va de
              oeste a este y de norte a sur
              Si el paso de malla no es congruente con los limites impuestos
              se ajusta para que lo sea al paso mas cercano. Se lanza un
              warning avisando de la situacion

Ejemplo:    [puntos,nFil,nCol]=geo_CreaMallaPuntosElipsoide(latNW,lonNW,latSE,
            lonSE,pasoLat,pasoLon);
            [puntos,nFil,nCol]=geo_CreaMallaPuntosElipsoide(latNW,lonNW,latSE,
```

```
            lonSE,pasoLat,pasoLon,hNW,hSE);

  Usa geoVarGlobal: Si

  Nota:       El sentido de la malla creda en longitud sera siempre de oeste a
              este, entendiendo como oeste la longitud de inicio y como este la
              de fin. Una malla de 5 puntos que vaya desde la longitud 170 a la
              -170 pasara por los puntos [170 175 180 -175 -170] mientras que
              una que vaya desde la longitud -170 a 170 pasara por los puntos
              [-170 -85 0 85 170]

  Original:   15-12-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: 19-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Reorganizacion de la ayuda
                Reescritura de la funcion para que no utilice la variable
                varargin como contenedor de los argumentos de entrada
                Adicion de sentencias de test
```

- **geo_CreaMallaPuntosPlano.m**:

```
  Funcion:    geo_CreaMallaPuntosPlano.m

  Proposito:  Crea una malla de puntos coordenados sobre el plano

  Entradas:   - Coordenada X del punto de la esquina superior izquierda
              - Coordenada Y del punto de la esquina superior izquierda
              - Coordenada X del punto de la esquina inferior derecha
              - Coordenada Y del punto de la esquina inferior derecha
              - Paso de malla para la coordenada X, en valor absoluto
              - Paso de malla para la coordenada Y, en valor absoluto
              Argumentos opcionales:
                - Coordenada Z del punto de la esquina superior izquierda
                - Coordenada Z del punto de la esquina inferior derecha

  Salidas:    - Matriz con la siguiente estructura:
                - Col. 1: Coordenada X de cada punto
                - Col. 2: Coordenada Y de cada punto
                Si se ha indicado coordenada Z:
                  - Col. 3: Coordenada Z de cada punto. El paso en Z se calcula
                            a partir del numero de puntos y de la diferencia
                            entre la Z de la esquina inferior derecha menos la
                            de la esquina superior izquierda
              - Numero de filas de la malla creada (puntos en Y)
              - Numero de columnas de la malla creada (puntos en X)
                Los puntos de la malla se representan de izquierda a derecha y
                de arriba a abajo. Es decir, la malla:
                                                [x1 y1 |z1|
                1  2                             x2 y2 |z2|
                3  4  queda representada como    x3 y3 |z3|
                5  6                             x4 y4 |z4|
                                                 x5 y5 |z5|
                                                 x6 y6 |z6|]
                Si se ha solicitado cÃąlculo de coordenada Z, esta tambien va de
                izquierda a derecha y de arriba a abajo
                Si el paso de malla no es congruente con los limites impuestos
                se ajusta para que lo sea al paso mas cercano. Se lanza un
                warning avisando de la situacion
```

```
Ejemplo:    [puntos,nFil,nCol]=geo_CreaMallaPuntosPlano(xNW,yNW,xSE,ySE,pasoX,
            pasoY);
            [puntos,nFil,nCol]=geo_CreaMallaPuntosPlano(xNW,yNW,xSE,ySE,pasoX,
            pasoY,zNW,zSE);

Usa geoVarGlobal: Si

Original:   15-12-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 19-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_DesactivaControles.m**:

```
Funcion:    geo_DesactivaControles.m

Proposito:  Desactiva los controles indicados como argumentos de entrada

Entradas:   - Ningun argumento o una serie de cadenas de texto con el nombre
              del tipo de control a desactivar. Las cadenas de texto
              permitidas son (en mayusculas o minusculas):
              - CONDICIONES: Desactiva el control de condiciones especiales
              - DIMENSIONES: Desactiva el control de dimensiones de los datos
              - DOMINIO: Desactiva el control de dominio de dato
              - TIPODATO: Desactiva el control de tipo de dato
              - TIEMPOFICHERO: Desactiva el control de tiempo en fichero
              - TIEMPOPANTALLA: Desactiva el control de tiempo en pantalla

Salidas:    - Desactiva los controles pedidos en la variable geoVarGlobal
            - Estructura con el estado previo de los controles (solo existen
              los campos correspondientes a los controles que han sido
              desactivados):
              - previo.CONDICIONES
              - previo.DIMENSIONES
              - previo.DOMINIO
              - previo.TIPODATO
              - previo.TIEMPOFICHERO
              - previo.TIEMPOPANTALLA
              Hay un campo que existe siempre y que indica si se ha
              desactivado algun control (0/1 -> no/si):
              - previo.CAMBIOS

Ejemplo:    previo=geo_DesactivaControles;
            previo=geo_DesactivaControles('CONDICIONES','DOMINIO',...);

Usa geoVarGlobal: Si

Nota:       Esta funcion no controla internamente el tipo de dato pasado

Original:   07-06-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 25-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

- **geo_DominioAngular.m**:

```
Funcion:    geo_DominioAngular.m

Proposito:  Reduce un valor angular al dominio ]-2*pi, 2*pi[

Entradas:   - Un valor angular en radianes, escalar o vector columna

Salidas:    - Angulos reducidos en radianes, escalar o vector columna

Ejemplo:    anguloBien=geo_DominioAngular(anguloMal);

Usa geoVarGlobal: No

Original:   12-08-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 19-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            04-03-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Simplificacion del codigo mediante el uso de la funcion rem()
```

- **geo_ExisteTextoEnFichero.m**:

```
Funcion:    geo_ExisteTextoEnFichero.m

Proposito:  Comprueba si una cadena de texto existe en un fichero

Entradas:   - Cadena de texto a buscar
            - Direccion completa del fichero, como cadena de texto
            Argumentos opcionales:
              - Indicador de busqueda de todas las ocurrencias de la cadena
                pasada o solo de la primera, dos posibles valores:
                - 1, para buscar todas las ocurrencias
                - 0, para buscar solo la primera ocurrencia
                Por defecto, este argumento vale 1

Salidas:    - Indicador de si se ha encontrado alguna coincidencia:
              - 0, si no se ha encontrado
              - Distinto de 0, si se ha encontrado. Numero de lineas en el
                fichero donde ha habido al menos una coincidencia
            - Matriz de tres columnas que almacena informacion sobre las
              cadenas encontradas. Por cada linea del fichero en la que se
              encuentre la cadena se almacenan:
              - Col. 1: Numero de linea en la que se ha encontrado la cadena
              - Col. 2: Numero de cadenas iguales a la pasada que hay en la
                        linea
              - Col. 3: Posicion en la linea en la que comienza la primera
                        ocurrencia de la cadena buscada
              Si no se ha encontrado el texto en el fichero, este argumento es
              una matriz vacia de 0x3

Ejemplo:    [indicador,posiciones]=geo_ExisteTextoEnFichero(texto,fichero);
            [indicador,posiciones]=geo_ExisteTextoEnFichero(texto,fichero,1);
            [indicador,posiciones]=geo_ExisteTextoEnFichero(texto,fichero,0);
```

```
Usa geoVarGlobal: Si

Original:   11-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geo_ExtraeAyudaFichero.m**:

```
Funcion:    geo_ExtraeAyudaFichero.m

Proposito:  Extrae la ayuda de una funcion (*.m), un fichero de datos (*.dat),
            un fichero C (*.c o *.h), un fichero C++ (*.cc o *.h), un fichero
            Fortran (*.f) o un script de Bash (*.sh) de GEO

Entradas:   - Direccion completa del fichero a leer, como cadena de texto

Salidas:    - Indicador de si el fichero pasado es un tipo de fichero conocido
              por GEO (*.m, *.dat, *.c, *.cc, *.h, *.f o *.sh):
              - 1, es un fichero conocido
              - 0, es un fichero desconocido
            - Cadena de texto que contiene la ayuda principal de la funcion
              (instrucciones de uso). Esta cadena de texto no termina con
              nueva linea ni retorno de carro. Si la funcion no contiene
              ayuda, este argumento es una cadena vacia
            Con tres argumentos de salida:
              - Cadena de texto que contiene la ayuda secundaria de la funcion
                (funciones utilizadas y relacionadas de GEO). Esta cadena de
                texto no termina con nueva linea ni retorno de carro. Si la
                funcion no contiene ayuda secundaria, este argumento es una
                cadena vacia
            Con cuatro argumentos de salida:
              - Cell array que contiene los nombres de las funciones
                utilizadas de GEO en la funcion pasada. Si la funcion no
                utiliza ninguna otra funcion de GEO, este argumento es un cell
                array vacio
              - Cell array que contiene los nombres de las funciones
                relacionadas de GEO con la funcion pasada. Si la funcion no
                tiene ninguna otra funcion relacionada de GEO, este argumento
                es un cell array vacio

Ejemplo:    [tipo,ayuda]=geo_ExtraeAyudaFichero(fichero);
            [tipo,ayuda,funciones]=geo_ExtraeAyudaFichero(fichero);
            [tipo,ayuda,funUtil,funRel]=geo_ExtraeAyudaFichero(fichero);

Usa geoVarGlobal: Si

Original:   11-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 11-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Adicion de la capacidad de extraccion de ayuda en formato
              #define
```

- **geo_Fecha.m**:

```
Funcion:    geo_Fecha.m
```

```
Proposito:  Calcula la fecha y hora del instante actual

Entradas:   Nada

Salidas:    - Cadena de texto en la forma 'dd-mm-aaaa, hr:min:seg', que
              contiene la fecha y hora del instante en que ha sido llamada la
              funcion

Ejemplo:    fecha=geo_Fecha;

Usa geoVarGlobal: No

Original:   02-09-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 20-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias para la comprobacion del numero de
              argumentos de entrada
              Adicion de sentencias de test
```

- **geo_FicheroConocido.m**:

```
Funcion:    geo_FicheroConocido.m

Proposito:  Comprueba si un fichero pasado es de algun tipo reconocido por
            GEO: *.m, *.dat, *.c, *.cc, *.h, *.f o *.sh

Entradas:   - Nombre del fichero a comprobar, como cadena de texto

Salidas:    - Indicador de si el fichero pasado es un tipo de fichero conocido
              por GEO (*.m, *.dat, *.c, *.cc, *.h, *.f o *.sh):
              - 1, es un fichero conocido
              - 0, es un fichero desconocido
            - Cadena de texto que contiene el caracter de comentario en el
              tipo de fichero correspondiente. Si el tipo de fichero no es
              conocido, este argumento es una cadena vacia
            - Cadena de texto que contiene la extension del fichero pasado
              Este argumento contiene la extension aunque el tipo de fichero
              no sea reconocido por GEO. Si el fichero pasado no tiene
              extension, este argumento es una cadena vacia

Ejemplo:    [tipo,carComentario]=geo_FicheroConocido(fichero);

Usa geoVarGlobal: Si

Nota:       La cadena de texto que informa de la extension comienza con el
            caracter '.', por ejemplo, en 'fichero.ext' la extension es
            '.ext', tal y como se almacena en la variable global
            geoVarGlobal.fichero.extension*

Original:   11-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 11-01-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Adicion de la extension como argumento de salida
```

- **geo_Find.m**:

```
Funcion:    geo_Find.m

Proposito:  Sustituta de la funcion find() de Octave/MatLab para el caso de
            busqueda en un vector columna. Esta funcion se ha programado
            porque la funcion find() en algunas versiones antiguas de
            GNU Octave (la 2.1.57 es una de ellas), cuando la ejecutamos sobre
            un escalar y no encuentra nada, devuelve una variable de
            dimensiones [1 0]. Como posteriormente se controla si la funcion
            ha devuelto algo por medio del analisis de las dimensiones de la
            salida, podriamos incurrir en un error al tomar el valor 1 de las
            filas cuando en realidad no hay nada

Entradas:   - Escalar o vector columna sobre el que actuara la funcion find()
              Ver la ayuda de dicha funcion

Salidas:    - Resultado de la ejecucion de find() en un formato tal que si es
              un argumento vacio, este tendra 0 filas
            - Numero de filas del resultado de la ejecucion de find()

Ejemplo:    [encontrado,filEncontrado]=geo_Find(vectorColumna);

Usa geoVarGlobal: No

Original:   01-12-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 19-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Eliminada una comprobacion, que se hacia mediante la funcion
              islogical(), porque era redundante y porque islogical() no
              existe en versiones antiguas de GNU Octave
              Adicion de sentencias de test
```

- **geo_FuncionesCodePad.m**:

```
Funcion:    geo_FuncionesCodePad.m

Proposito:  Escribe nombres de una lista de funciones en un fichero en el
            formato adecuado para anyadir a un archivo de definicion de
            sintaxis del editor de texto CodePad

Entradas:   - Un cell array de cadenas de texto que contiene el nombre de
              todas las funciones a escribir en el fichero
            - FontID del grupo de funciones a crear, como numero natural. Ver
              el fichero de definicion de sintaxis de CodePad (geoOctave.def)
              para obtener mas informacion acerca de este parametro
            - Nombre identificador del grupo de funciones a crear, como cadena
              de texto
            - Nombre del fichero de salida

Salidas:    - Numero de funciones
            - Un fichero que contiene que contiene un 'item' con el nombre de
              todas las funciones pasadas para ser anyadido a un fichero de
              definicion de sintaxis del editor CodePad

Ejemplo:    nF=geo_FuncionesCodePad(nombres,FontID,identificador,fichSalida);
```

```
Usa geoVarGlobal: Si

Original:   28-04-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_FuncionesGeo.m**:

```
Funcion:    geo_FuncionesGeo.m

Proposito:  Extrae el nombre de todas las funciones y/o ficheros de GEO

Entradas:   Dos posibilidades:
              - Ningun argumento de entrada
              - Cadena de texto 'TODO'

Salidas:    - Un cell array de cadenas de texto que contiene el nombre de
              todas las funciones de GEO. Dos posibilidades:
              - Si no se ha pasado ningun argumento de entrada, cell array de
                cadenas de texto con los nombres de las funciones sin
                extension, ordenadas alfabeticamente
              - Si se ha pasado un argumento de entrada, cell array de cadenas
                de texto con los nombres de todos los ficheros de GEO
                (funciones, ficheros de datos y ficheros C/C++, Fortran y
                Bash), ordenados alfabeticamente. En este caso el nombre
                contiene la ruta completa de los ficheros junto con su
                extension. Entran los siguientes tipos de fichero:
                - Funciones de GNU Octave: todas, tanto funciones como
                                           programas de prueba
                - Ficheros compilables: todos
                - Ficheros Bash: todos
            - Numero de funciones o ficheros

Ejemplo:    [fun,nFun] = geo_FuncionesGeo;
            [fich,nFich] = geo_FuncionesGeo('TODO');

Usa geoVarGlobal: Si

Original:   31-05-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 25-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_FuncionesGeoResaltado.m**:

```
Funcion:    geo_FuncionesGeoResaltado.m

Proposito:  Extrae el nombre de todas las funciones de GEO a un fichero en el
            formato adecuado para anadir a un archivo de definicion de
            sintaxis de los editores de texto que soporten a GEO
```

```
Entradas:   Nada

Salidas:    - Numero de funciones
            - Ficheros de sintaxis para cada editor de texto que soporte a GEO
              con la lista de las funciones de la biblioteca preparada para
              ser anyadida al fichero general de sintaxis correspondiente

Ejemplo:    nFunciones=geo_FuncionesGeoResaltado;

Usa geoVarGlobal: Si

Original:   28-04-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de la extraccion de funciones para el editor nano
              Adicion de sentencias de test
```

- **geo_FuncionesKate.m**:

```
Funcion:    geo_FuncionesKate.m

Proposito:  Escribe nombres de una lista de funciones en un fichero en el
            formato adecuado para anyadir a un archivo de definicion de
            sintaxis de los editores de texto Kate y KWrite

Entradas:   - Un cell array de cadenas de texto que contiene el nombre de
              todas las funciones a escribir en el fichero
            - Nombre identificador del grupo de funciones a crear, como cadena
              de texto
            - Nombre del fichero de salida

Salidas:    - Numero de funciones
            - Un fichero que contiene que contiene una lista de items con el
              nombre de todas las funciones pasadas para ser anyadida a un
              fichero de definicion de sintaxis de los editores Kate y KWrite,
              del entorno de escritorio KDE

Ejemplo:    nFunciones=geo_FuncionesKate(nombres,identificador,fichSalida);

Usa geoVarGlobal: Si

Original:   28-04-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_FuncionesNano.m**:

```
Funcion:    geo_FuncionesNano.m

Proposito:  Escribe nombres de una lista de funciones en un fichero en el
            formato adecuado para anyadir a un archivo de definicion de
```

```
            sintaxis del editor de texto nano

  Entradas:   - Un cell array de cadenas de texto que contiene el nombre de
                todas las funciones a escribir en el fichero
              - Nombre del color para presentar en pantalla las funciones (ha de
                ser un color aceptado por nano, aunque esto no se controla en
                esta funcion), como cadena de texto
              - Nombre del fichero de salida

  Salidas:    - Numero de funciones
              - Un fichero que contiene una lista 'color' con el nombre de todas
                las funciones pasadas para ser anyadida a un fichero de
                definicion de sintaxis del editor nano

  Ejemplo:    nFunciones=geo_FuncionesNano(nombres,color,fichSalida);

  Usa geoVarGlobal: Si

  Original:   26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: Ninguna
```

- **geo_FuncionesVim.m**:

```
  Funcion:    geo_FuncionesVim.m

  Proposito:  Escribe nombres de una lista de funciones en un fichero en el
              formato adecuado para anyadir a un archivo de definicion de
              sintaxis del editor de texto vim

  Entradas:   - Un cell array de cadenas de texto que contiene el nombre de
                todas las funciones a escribir en el fichero
              - Nombre identificador del grupo de funciones a crear, como cadena
                de texto
              - Nombre del fichero de salida

  Salidas:    - Numero de funciones
              - Un fichero que contiene que contiene una lista 'syn keyword' con
                el nombre de todas las funciones pasadas para ser anyadida a un
                fichero de definicion de sintaxis del editor vim

  Ejemplo:    nFunciones=geo_FuncionesVim(nombres,identificador,fichSalida);

  Usa geoVarGlobal: Si

  Original:   28-04-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

  Revisiones: 26-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
                Reorganizacion de la ayuda
                Reescritura de la funcion para que no utilice la variable
                varargin como contenedor de los argumentos de entrada
                Adicion de sentencias de test
```

- **geo_HayQueCalcularErrores.m**:

```
  Funcion:    geo_HayQueCalcularErrores.m
```

```
Proposito:  Determina si hay que calcular la propagacion de errores en una
            funcion

Entradas:   - Indicadores de la varianza-covarianza de los datos del problema

Salidas:    - 1: Hay que hacer el calculo de la propagacion de errores (al
                 menos un elemento de entrada es distinto de -1)
            - 0: No hay que hacer el calculo de la propagacion de errores
                 (todos los elementos de entrada son iguales a -1)

Ejemplo:    calError=geo_HayQueCalcularErrores(var1,var2,...,varN);

Usa geoVarGlobal: No

Original:   08-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 20-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Adicion de sentencias de test
```

- **geo_IgualesDistintoDeValor.m**:

```
Funcion:    geo_IgualesDistintoDeValor.m

Proposito:  Comprueba si todos los elementos en un vector columna distintos de
            un valor dado son iguales entre si

Entradas:   - Vector columna
            - Valor numerico

Salidas:    - 1, si todos los elementos del vector distintos del valor
              introducido como segundo argumento son iguales entre si
            - 0, en caso contrario

Ejemplo:    correcto=geo_IgualesDistintoDeValor(vector,valor);

Usa geoVarGlobal: Si

Original:   25-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geo_ImprimeTiempo.m**:

```
Funcion:    geo_ImprimeTiempo.m

Proposito:  Imprime en un fichero y/o en pantalla datos referentes al sistema
            operativo y al programa utilizado, asi como el tiempo empleado en
            la ejecucion de un proceso
            Las impresiones se realizan si las variables de control de
            geoVarGlobal lo permiten

Entradas:   - Nombre del fichero a generar, como cadena de texto, o indicador
              de fichero abierto para escribir. Dos posibilidades:
              - Si se pasa una cadena de texto como nombre del fichero y este
                no existe, se crea. Si existe se anyade la informacion pasada
                al contenido existente
```

```
                - Si se pasa un indicador de fichero abierto para escribir, se
                  anyade la informacion pasada al contenido existente a partir
                  de la posicion actual del puntero de L/E
              - Titulo del proceso en cuestion (programa, etc.), como cadena de
                texto
              - Cantidad de tiempo a imprimir, en segundos
              Argumentos opcionales:
                - Numero de saltos de linea y retornos de carro a escribir
                  previamente a la impresion de la informacion de tiempo. Por
                  defecto este parametro vale 0
                  Este argumento solo afecta a la escritura en fichero, no a la
                  escritura de la informacion en pantalla

Salidas:      - Indicador de escritura en pantalla: 0/1 -> no/si
              - Indicador de escritura en fichero: 0/1 -> no/si
              - Escritura en los medios indicados de los datos descritos. En la
                escritura en fichero hay dos posibilidades:
                - Si se ha pasado como primer argumento una cadena de texto, el
                  fichero donde se escribe se cierra al finalizar la ejecucion
                  de la funcion
                - Si se ha pasado como primer argumento un indicador de fichero
                  abierto para escribir, este no se cierra y el puntero de L/E
                  se queda en la ultima posicion escrita, no se produce un salto
                  de linea ni retorno de carro

Ejemplo:      [escPant,escFich]=geo_ImprimeTiempo(nombreFichero,titulo,tiempo);
              [escPant,escFich]=geo_ImprimeTiempo(idFichero,titulo,tiempo);
              [escPant,escFich]=geo_ImprimeTiempo(nombreFichero,titulo,tiempo,
              lineasPrevias);
              [escPant,escFich]=geo_ImprimeTiempo(idFichero,titulo,tiempo,
              lineasPrevias);

Usa geoVarGlobal: Si

Nota 1:       Las impresiones solo se realizan si las variables de control
              correspondientes de geoVarGlobal lo permiten:
                geoVarGlobal.varControl.tiempoPantalla==1
                geoVarGlobal.varControl.tiempoFichero==1

Nota 2:       En el caso de que ocurra un error y se haya pasado como primer
              argumento de entrada un indicador de fichero abierto, este
              permanecera abierto, aunque puede que el puntero de L/E no
              permanezca en su posicion original

Original:     01-09-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 20-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_IndGeoVarGlobal.m**:

```
Funcion:      geo_IndGeoVarGlobal.m

Proposito:    Obtiene el indice en la estructura 'geoVarGlobal' del tipo de
              variable introducida
```

```
Entradas:   - Nombre de la variable a localizar, como cadena de texto

Salidas:    - Indice, i, de la variable en la estructura geoVarGlobal, de
              forma que geoVarGlobal.var(i) contiene los datos asociados
              al tipo de variable
            - Un error, en el caso de que la variable no exista

Ejemplo:    indice=geo_IndVarGlobal('ACIMUT');

Usa geoVarGlobal: Si

Original:   23-15-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 20-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LeeDirectorio.m**:

```
Funcion:    geo_LeeDirectorio.m

Proposito:  Devuelve el nombre de los elementos incluidos en un directorio

Entradas:   - Nombre completo del directorio a leer, como cadena de texto

Salidas:    - Un cell array de cadenas de texto que contiene el nombre de
              todos los elementos cuyo nombre no comience por el caracter '.'
              contenidos en el directorio

Ejemplo:    contenido=geo_LeeDirectorio(nombreDirectorio);

Usa geoVarGlobal: Si

Nota:       Los elementos cuyo nombre comienza por el caracter '.' son
            elementos ocultos en un sistema UNIX

Original:   05-02-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 20-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Se eliminan de la salida los elementos cuyo nombre empiece por
              el caracter '.' (ocultos en UNIX)
              Adicion de sentencias de test
```

- **geo_LeeLineaFichero.m**:

```
Funcion:    geo_LeeLineaFichero.m

Proposito:  Lee una linea de un fichero

Entradas:   - Identificador de fichero abierto para leer con el puntero de
              lectura/escritura, (L/E), colocado al principio de la linea a
```

```
                extraer
              - El nombre del fichero a leer, como cadena de texto
              - El numero de la linea a leer, como numero natural. Este
                parametro solo se utiliza para dar informacion en los posibles
                mensajes de error, en ningun momento se utiliza en el proceso de
                lectura
              - Una geoLista que contiene la definicion de los tipos de dato que
                contiene la linea a leer. La geoLista estara compuesta por
                estructuras que contienen los campos 'longitud' y 'tipo', cuyos
                valores son los correspondientes a los mismos campos de la
                variable global
                    geoVarGlobal.var().longitud
                    geoVarGlobal.var().tipo
                De esta forma, se definen los componentes de la linea a leer
                Con estos datos la funcion calcula la longitud de la linea a
                leer

Salidas:      - Una geoLista, compuesta por los elementos de la linea leida si
                todo se ha ejecutado correctamente. Se dan las siguientes
                excepciones:
                - Una geoLista vacia si el puntero de L/E estaba posicionado en
                  la ultima posicion del fichero y no se ha podido leer nada
                - Un error, en cualquier otro caso
              - Numero de elementos leidos correctamente

Ejemplo:      numLinea=25;
              campo1.longitud=geoVarGlobal.var(indice1).longitud;
              campo1.tipo=geoVarGlobal.var(indice1).tipo;
              campo2.longitud=geoVarGlobal.var(indice2).longitud;
              campo2.tipo=geoVarGlobal.var(indice2).tipo;
              lista=geo_Lista(campo1,campo2);
              [linea,nElem]=geo_LeeLineaFichero(idFich,nomFich,numLinea,lista);

Usa geoVarGlobal: Si

Nota:         En el caso de que ocurra un error, el fichero pasado como primer
              argumento permanecera abierto, aunque la posicion original del
              puntero de L/E puede no ser la misma que la original

Original:     24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 21-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Modificacion de la funcion para que las cadenas de texto leidas
              no sean pasadas a mayusculas
              Modificacion de la funcion para que tambien devuelva el numero
              de elementos leidos correctamente
            06-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Corregido error que hacia que la funcion no indicase error en el
              caso de leer una cadena de texto cuando se espera un dato
              numerico
```

- **geo_Path.m**:

```
Funcion:    geo_Path.m

Proposito:  Calcula la ruta de todas las carpetas de GEO

Entradas:   Nada

Salidas:    - Un cell array de cadenas de texto que contiene las rutas de
              todas las carpetas de GEO ordenadas alfabeticamente
            - Numero de rutas

Ejemplo:    [rutas,nRutas]=geo_Path;

Usa geoVarGlobal: Si

Original:   03-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 20-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_PosFicheroLineaDatos.m**:

```
Funcion:    geo_PosFicheroLineaDatos.m

Proposito:  Posiciona el puntero de L/E en un fichero al inicio de la primera
            linea que contenga datos, es decir, en la primera linea que no sea
            un comentario o este vacia (tambien se entiende por linea vacia
            aquella que solo contiene espacios en blanco)

Entradas:   - Indicador de fichero abierto para leer
            - Cadena que contiene el caracter de comentario a utilizar

Salidas:    - Numero de linea donde se ha posicionado el puntero de L/E a
              partir de su posicion original (al inicio de la funcion). En el
              caso de que todas las lineas del fichero sean comentarios o
              esten vacias, este argumento vale -1 y el puntero de L/E se
              posiciona en el lugar donde estaba al principio de la funcion
              Si la linea original es valida, el puntero no se movera de sitio
              y este argumento valdra 0
              El fichero permanecera abierto al salir de la funcion

Ejemplo:    posicion=geo_PosFicheroLineaDatos(idFichero,'caracterComentario');

Usa geoVarGlobal: Si

Nota:       En el caso de que ocurra un error, el fichero permanecera abierto,
            aunque la posicion del puntero de L/E puede no ser la misma que la
            original

Original:   26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geo_QuitaEspacios.m**:

```
Funcion:    geo_QuitaEspacios.m
```

```
Proposito:  Elimina los espacios en blanco de una cadena de texto. Los
            espacios en blanco son todos aquellos caracteres para los cuales
            'isspace()' devuelve el valor 1

Entradas:   - Cadena de texto
            Argumentos opcionales:
              - Cadena de texto identificadora del tipo de espacios a eliminar
                Las opciones son:
                'TODOS': Elimina todos los espacios en blanco de la cadena
                         pasada
                'PRINCIPIO': Elimina los espacios en blando situados al
                             principio de la cadena pasada
                'FINAL': Elimina los espacios en blando situados al final de
                         la cadena pasada
                'PRINCIPIO-FINAL': Elimina los espacios en blando situados al
                                   principio y al final de la cadena pasada,
                                   respetando aquellos situados entre palabras
                                   o caracteres
            En el caso de llamar a la funcion con un solo argumento, el
            identificador de tipo de espacios a eliminar es 'TODOS'

Salidas:    - La cadena pasada, sin los espacios especificados

Ejemplo:    cadena=geo_QuitaEspacios(' acimut geodesico ');
            cadena=geo_QuitaEspacios(' acimut geodesico ','TODOS');
            cadena=geo_QuitaEspacios(' acimut geodesico ','PRINCIPIO');
            cadena=geo_QuitaEspacios(' acimut geodesico ','FINAL');
            cadena=geo_QuitaEspacios(' acimut geodesico ','PRINCIPIO-FINAL');

Usa geoVarGlobal: Si

Original:   30-05-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 21-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_RestauraControles.m**:

```
Funcion:    geo_RestauraControles.m

Proposito:  Restaura el valor original de los controles modificados por las
            funciones geo_ActivaControles() y geo_DesactivaControles()

Entradas:   - La estructura devuelta por las funciones
              geo_ActivaControles() o geo_DesactivaControles()

Salidas:    - El valor de los controles modificados restaurados a su valor
              original

Ejemplo:    geo_RestauraControles(previo);

Usa geoVarGlobal: Si

Nota:       Esta funcion no controla internamente el tipo de dato pasado
```

```
Original:   07-06-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 25-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Eliminacion de la variable de salida 'estado', ya que era inutil
              'estado' solo podia valer 1 ya que en caso de error la ejecucion
              de la funcion se aborta y no devuelve nada
              Adicion de sentencias de test
```

- **geo_Subcadena.m**:

```
Funcion:    geo_Subcadena.m

Proposito:  Extrae una porcion de una cadena de texto pasada como argumento

Entradas:   - Cadena de texto
            - Posicion de la cadena de entrada donde comienza la extraccion,
              numero entero distinto de 0. Si el numero es negativo, la
              extraccion comienza desde el final
            Argumentos opcionales:
              - Longitud de la cadena de salida, numero natural
              Si este argumento de omite, la cadena de salida extraida va
              desde la posicion indicada en el segundo argumento hasta el
              final de la cadena original

Salidas:    - La subcadena definida por los parametros correspondientes

Ejemplo:    subcadena=geo_Subcadena('geodesia',2,5) -> 'eodes'
            subcadena=geo_Subcadena('geodesia',-7,5) -> 'eodes'

Usa geoVarGlobal: Si

Original:   24-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 21-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_TestFuncionesGeo.m**:

```
Funcion:    geo_TestFuncionesGeo.m

Proposito:  Ejecuta las sentencias de test escritas en las funcion de GEO

Entradas:   Nada

Salidas:    - Numero de funciones de GEO, tanto si se ha solicitado como si no
              Tambien se muestra informacion en pantalla sobre los test que se
              van haciendo y los posibles errores que ocurran
            - Numero de funciones en las que ha fallado algun test, solo si se
              ha solicitado. En este caso solo se muestran en pantalla los
              posibles errores que ocurran
```

```
            - Cell array que contiene los nombres de las funciones en las que
              ha fallado algun test, solo si se ha solicitado. Si no ha
              fallado ningun test este argumento es un cell array vacio. En
              este caso solo se muestran en pantalla los posibles errores que
              ocurran

Ejemplo:    [nFun]=geo_TestFuncionesGeo;
            [nFun,nFallos]=geo_TestFuncionesGeo;
            [nFun,nFallos,funFallos]=geo_TestFuncionesGeo;

Usa geoVarGlobal: Si

Original:   25-10-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geo_VarGeoVarGlobal.m**:

```
Funcion:    geo_VarGeoVarGlobal.m

Proposito:  Extrae la estructura correspondiente al dato de indice (i) de
            'geoVarGlobal'

Entradas:   - Indice de la variable en 'geoVarGlobal', como numero entero

Salidas:    - La estructura correspondiente a geoVarGlobal.var(i), que
              contiene los datos asociados al tipo de variable. Los campos
              son:
              - estructura.nombre
              - estructura.tipo
              - estructura.longitud
              - estructura.unidadesGeodesia
              - estructura.unidadesNaturales
              - estructura.dominio
              - estructura.limitesDominio
              - estructura.dominioCasosEspeciales
              - estructura.limitesDominioCasosEspeciales

Ejemplo:    estructura=geo_VarGeoVarGlobal(16);

Usa geoVarGlobal: Si

Original:   23-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 21-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_WarningOct.m**:

```
Funcion:    geo_WarningOct.m

Proposito:  Lanza un mensaje de warning cuando una funcion *.oct falla

Entradas:   - Nombre de la funcion para la que se invoca el warning, como
```

```
                cadena de texto
              - Nombre de la funcion *.oct que ha fallado, como cadena de texto
              - Nombre de la funcion que se ejecutara en lugar de la funcion
                *.oct, como cadena de texto
              Argumento opcional:
                - Mensaje de error, como cadena de texto. Este argumento deberia
                  ser el contenido de la variable 'lasterr' de GNU Octave y
                  MatLab

Salidas:      - Mensaje de warning en pantalla

Ejemplo:      geo_WarningOct(nomFuncion,nomFuncOct,nomFuncM);
              geo_WarningOct(nomFuncion,nomFuncOct,nomFuncM,lasterr);

Usa geoVarGlobal: Si

Original:     21-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 02-11-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

## 10.18. Carpeta 18-nuevas

En esta carpeta se almacenan las funciones de GEO que están en desarrollo, pero todavía no son lo suficientemente estables como para poder ser utilizadas con garantías.

## 10.19. Carpeta 19-viejas

En esta carpeta se almacenan las funciones de GEO que se consideran desfasadas porque hay nuevas funciones que realizan la misma tarea. Estas funciones solo se mantienen por motivos de compatibilidad con funciones antiguas y no es recomendable su uso.

- **geo_CambiaEstadoControles.m**:

```
Funcion:      geo_CambiaEstadoControles.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:    Cambia el estado de las variables de control en la estructura
              geoVarGlobal

Entradas:     - Nuevo valor para el control de tipo de dato (0 o 1)
              - Nuevo valor para el control de dominio de dato (0 o 1)
              - Nuevo valor para el control de dimensiones de dato (0 o 1)
              - Nuevo valor para el control de condiciones especiales (0 o 1)
              - Nuevo valor para el control de impresion de tiempo en pantalla
                (0 o 1)
              - Nuevo valor para el control de impresion de tiempo en fichero
                (0 o 1)

Salidas:      - Variable de estado que vale 1 si la funcion se ha ejecutado sin
                errores
```

```
            - Los campos de control de la estructura geoVarGlobal cambiados

  Ejemplo:    geo_CambiaEstadoControles(0,0,0,0,0,0);

  Funciones utilizadas de GEO: - geo_EsEstero.m

  Funciones relacionadas de GEO: - geo_EstadoControles.m

  Ultima revision: 01-09-07

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_EstadoControles.m**:

```
  Funcion:    geo_EstadoControles.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Informa del valor de las variables de control en la estructura
              geoVarGlobal

  Entradas:   Ninguna

  Salidas:    - Estructura con los siguientes campos:
                - controles.tipoDato: estado del control de tipo de dato
                - controles.dominio: estado del control de dominio de dato
                - controles.dimensiones: estado del control de dimensiones de
                                         dato
                - controles.condiciones: estado del control de condiciones
                                         especiales
                - controles.tiempoPantalla: estado del control de impresion de
                                            tiempo en pantalla
                - controles.tiempoFichero: estado del control de impresion de
                                           tiempo en fichero

  Ejemplo:    controles=geo_EstadoControles;

  Funciones utilizadas de GEO: Ninguna

  Funciones relacionadas de GEO: - geo_CambiaEstadoControles.m

  Ultima revision: 01-09-07

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_UdsTrabajoUdsBase.m**:

```
  Funcion:    geo_UdsTrabajoUdsBase.m

  Proposito:  Transforma valores a unidades base de GEO

  Entradas:   - Un escalar, vector o matriz
              - Numero identificador de las unidades de partida. Ha de coincidir
                con el indice de la estructura global geoVarGlobal.unidades{i}
                Este parametro indica a la funcion si la transformacion es
                lineal o angular
```

```
Salidas:    - El argumento de entrada transformado a unidades base. Ver la
              estructura global eoVarGlobal.unidadesBase

Ejemplo:    salida=geo_UdsTrabajoUdsBase(entrada,5);

Usa geoVarGlobal: Si

Original:   26-07-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 21-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_UdsTrabajoUdsBaseVar.m**:

```
Funcion:    geo_UdsTrabajoUdsBaseVar.m

Proposito:  Transforma valores a unidades base de varianza de GEO

Entradas:   - Un escalar, vector o matriz
            - Numero identificador de las unidades de varianza de partida. Ha
              de coincidir con el indice de la estructura global
              geoVarGlobal.unidadesVar{i}
              Este parametro indica a la funcion si la transformacion es
              lineal o angular

Salidas:    - El argumento de entrada transformado a unidades base de varianza
              Ver la estructura global eoVarGlobal.unidadesBaseVar

Ejemplo:    salida=geo_UdsTrabajoUdsBaseVar(entrada,5);

Usa geoVarGlobal: Si

Original:   26-07-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 21-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

# Capítulo 11
# Sección 03-elipsoide

En esta carpeta se almacenan las funciones relacionadas con el trabajo con elipsoides.

- **geo_Elipsoide.m**:

```
Funcion:    geo_Elipsoide.m

Proposito:  Crea una estructura 'elipsoide'

Entradas:   - Codigo del elipsoide, como cadena de texto. No se hace
              distincion entre mayusculas y minusculas

Salidas:    - Una estructura 'elipsoide', cuyos campos son:
              - codigo:     Codigo del elipsoide, como cadena de texto
              - a:          Semieje mayor del elipsoide, en metros
              - b:          Semieje menor del elipsoide, en metros
              - f:          Aplanamiento del elipsoide: (a-b)/a
              - priExc2:    Primera excentricidad al cuadrado, (a^2-b^2)/a^2
              - segExc2:    Segunda excentricidad al cuadrado, (a^2-b^2)/b^2
              - paramDin:   Indicador de existencia de parametros dinamicos
                            Dos posibilidades:
                            - 0: No
                            - 1: Si
                            Esta funcion todavia no soporta la existencia de
                            parametros dinamicos, por lo que este campo
                            siempre valdra 0
              - comentario: Comentario acerca del elipsoide

Ejemplo:    elipsoide=geo_Elipsoide('in');

Usa geoVarGlobal: Si

Original:   28-01-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 13-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_EsElipsoide.m**:

```
Funcion:    geo_EsElipsoide.m

Proposito:  Comprueba si el argumento pasado es una estructura 'elipsoide'
```

```
Entradas:   - Una variable

Salidas:    - 1, si el argumento de entrada es una estructura 'elipsoide'
            - Un error, en caso contrario

Ejemplo:    esElipsoide=geo_EsElipsoide(elipsoide);

Usa geoVarGlobal: Si

Original:   28-01-2007, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 13-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

# Capítulo 12
# Sección 04-geoide

En esta carpeta se almacenan las funciones relacionadas con el trabajo con los geoides.

## 12.1. Carpeta 01-estructuraGeoide

En esta carpeta se almacenan las funciones relacionadas conla creación de la estructura geoide.

- **geo_EsGeoide.m**:

```
Funcion:    geo_EsGeoide.m

Proposito:  Comprueba si el argumento pasado es una estructura 'geoide'

Entradas:   - Una variable

Salidas:    - 1, si el argumento de entrada es una estructura 'geoide'
            - Un error, en caso contrario

Ejemplo:    esGeoide=geo_EsGeoide(geoide);

Usa geoVarGlobal: Si

Original:   04-05-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 08-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            10-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Adicion de la comprobacion del campo 'repPrimPto'
```

- **geo_Geoide.m**:

```
Funcion:    geo_Geoide.m

Proposito:  Crea una estructura 'geoide'

Entradas:   - Codigo del modelo de geoide, como cadena de texto. No se hace
              distincion entre mayusculas y minusculas

Salidas:    - Una estructura 'geoide', cuyos campos son:
              - codigo:      Codigo del geoide. Cadena de texto
```

```
- elipsoide:   Codigo del elipsoide al que se refiere el geoide
               Cadena de texto
- coorPuntos:  Indicador del sistema de coordenadas en que
               vienen expresados los puntos dato. Dos
               posibilidades:
               - 0: Coordenadas cartesianas
               - 1: Coordenadas geodesicas
- cobGlobLat:  Indicador de cobertura global en latitud. Dos
               posibilidades:
               - 0: No
               - 1: Si
- cobGlobLon:  Indicador de cobertura global en longitud. Dos
               posibilidades:
               - 0: No
               - 1: Si
- repPrimPto:  Indicador de repeticion del primer punto en la
               direccion este-oeste de cada fila de la malla de
               datos. Dos posibilidades:
               - 0: No se repite el primer punto
               - 1: Si se repite el primer punto
               Este argumento solo se tendra en cuenta si el
               modelo tiene cobertura global en la direccion
               este-oeste
- errores:     Indicador de informacion de errores. Dos
               posibilidades:
               - 0: No
               - 1: Si
- tipoDato:    Nombre del tipo de dato binario de GEO utilizado
               para almacenar los datos de ondulacion. Cadena de
               texto. Los posibles nombres de los tipos de dato
               binario se definen en
               geoVarGlobal.nombreDatoBinGeo{}
- dimDato:     Numero identificador de las dimensiones de los
               datos de ondulacion. Los posibles numeros
               indicadores de dimensiones se definen en
               geoVarGlobal.unidades{}
- tipoAlm:     Nombre del tipo de almacenamiento de los datos de
               ondulacion en el fichero binario correspondiente
               Los posibles nombres de los tipos de
               almacenamiento se definen en
               geoVarGlobal.tipoAlmacenBinarioGeo{}
- tipoDatoVar: Nombre del tipo de dato binario de GEO utilizado
               para almacenar los datos de varianza. Cadena de
               texto
- dimDatoVar:  Numero identificador de las dimensiones de los
               datos de varianza
- tipoAlmVar:  Nombre del tipo de almacenamiento de los datos de
               varianza en el fichero binario correspondiente
- noDato:      Indicador de existencia de puntos sin dato de
               ondulacion. Dos posibilidades:
               - 0: No
               - 1: Si
- idNoDato:    Valor identificativo de ausencia de dato de
               ondulacion. Numero entero
- noDatoVar:   Indicador de existencia de puntos sin dato de
               varianza. Dos posibilidades:
               - 0: No
```

```
                        - 1: Si
              - idNoDatoVar: Valor identificativo de ausencia de dato de
                        varianza. Numero entero
              - limNorte:    Limite norte de la malla de puntos (si el sistema
                        de coordenadas es geodesico, en radianes)
              - limSur:      Limite sur de la malla de puntos (si el sistema
                        de coordenadas es geodesico, en radianes)
              - limEste:     Limite este de la malla de puntos (si el sistema
                        de coordenadas es geodesico, en radianes)
              - limOeste:    Limite oeste de la malla de puntos (si el sistema
                        de coordenadas es geodesico, en radianes)
              - pasoLat:     Paso de malla en latitud (si el sistema de
                        coordenadas es geodesico, en radianes)
              - pasoLon:     Paso de malla en longitud (si el sistema de
                        coordenadas es geodesico, en radianes)
              - comentario:  Comentario acerca del modelo de geoide

Ejemplo:    geoide=geo_Geoide('egm96');

Usa geoVarGlobal: Si

Original:   04-05-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 06-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

## 12.2. Carpeta 02-ondulacionGeoide

En esta carpeta se almacenan las funciones de interpolación de ondulaciones en el geoide.

- **geo_OndGeoideMalla.m**:

```
Funcion:    geo_OndGeoideMalla.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la ondulacion del geoide a partir de una malla de puntos
            de ondulacion conocida y calcula una estimacion del error cometido
            (varianza)

Entradas:   - Latitud geodesica (en radianes) o coordenada X del punto de
              trabajo (escalar o vector columna)
            - Longitud geodesica (en radianes) o coordenada Y del punto de
              trabajo (escalar o vector columna)
            - Tamano de la ventana de interpolacion, 1 o numero par entre 2 y
              10. Si el numero es 1 se utiliza el vecino mas proximo como
              solucion, si es 2 se realiza una interpolacion bilineal. En otro
              caso, la interpolacion se realiza por medio de splines cubicos
            - Estructura geoide
            Argumentos opcionales:
              - Indicador de calculo de error, numero entero. Si este
                argumento toma el valor -1 o se omite no se realiza calculo
                del error. En otro caso, el calculo se lleva a cabo. Si se
                pide que calcule error y el modelo de geoide no dispone de
```

```
                informacion, el calculo no se realiza y la funcion lanza un
                warning

Salidas:    - Ondulacion del geoide (escalar o vector columna), en las mismas
              unidades que las de las ondulaciones dato de la malla de puntos
            - Varianza de la ondulacion calculada (escalar o vector columna),
              en las mismas unidades (al cuadrado) que las de la ondulacion
              En el caso de no haber pasado datos para el calculo del error
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [ond]=geo_OndGeoideMalla(latGeo,lonGeo,ventana,geoide);
            [ond]=geo_OndGeoideMalla(latGeo,lonGeo,ventana,geoide,-1);
            [ond,varOnd]=geo_OndGeoideMalla(latGeo,lonGeo,ventana,geoide,1);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_EsGeoide.m
                             - geo_ControlDominioInterpMalla.m
                             - geo_DesactivaControles.m
                             - geo_ExtraeVentanaMalla.m
                             - geo_WarningOct.m
                             - geo_InterpolaMalla.m
                             - geo_QuitaEspacios.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_AltitudMdeMalla.m

Ultima revision: 04-03-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

# Capítulo 13
# Sección 05-mde

En esta carpeta se almacenan las funciones relacionadas con el trabajo con los modelos digitales de elevaciones.

## 13.1. Carpeta 01-estructuraMde

En esta carpeta se almacenan las funciones relacionadas conla creación de la estructura MDE.

- **geo_EsMde.m**:

```
Funcion:    geo_EsMde.m

Proposito:  Comprueba si el argumento pasado es una estructura 'mde'

Entradas:   - Una variable

Salidas:    - 1, si el argumento de entrada es una estructura 'mde'
            - Un error, en caso contrario

Ejemplo:    esMde=geo_EsMde(mde);

Usa geoVarGlobal: Si

Original:   04-05-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 09-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
            10-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Adicion de la comprobacion del campo 'repPrimPto'
```

- **geo_Mde.m**:

```
Funcion:    geo_Mde.m

Proposito:  Crea una estructura 'mde'

Entradas:   - Codigo del modelo digital de elevaciones, como cadena de texto
              No se hace distincion entre mayusculas y minusculas

Salidas:    - Una estructura 'mde', cuyos campos son:
```

```
- codigo:       Codigo del MDE. Cadena de texto
- tipoAltitud:  Indicador del tipo de altitudes que almacena el
                MDE. Dos posibilidades:
                - 0: Altitud elipsoidal
                - 1: Altitud ortometrica
- supRef:       Codigo de la superficie de referencia a la que se
                refieren las altitudes. Cadena de texto
- elipsoide:    Codigo del elipsoide al que se refieren las
                coordenadas planimetricas de los puntos dato. No
                tiene por que coincidir con el posible elipsoide
                almacenado en el campo 'supRef'. Cadena de texto
- coorPuntos:   Indicador del sistema de coordenadas en que
                vienen expresados los puntos dato. Dos
                posibilidades:
                - 0: Coordenadas cartesianas
                - 1: Coordenadas geodesicas
- cobGlobLat:   Indicador de cobertura global en latitud. Dos
                posibilidades:
                - 0: No
                - 1: Si
- cobGlobLon:   Indicador de cobertura global en longitud. Dos
                posibilidades:
                - 0: No
                - 1: Si
- repPrimPto:   Indicador de repeticion del primer punto en la
                direccion este-oeste de cada fila de la malla de
                datos. Dos posibilidades:
                - 0: No se repite el primer punto
                - 1: Si se repite el primer punto
                Este argumento solo se tendra en cuenta si el
                modelo tiene cobertura global en la direccion
                este-oeste
- errores:      Indicador de informacion de errores. Dos
                posibilidades:
                - 0: No
                - 1: Si
- tipoDato:     Nombre del tipo de dato binario de GEO utilizado
                para almacenar los datos de altitud. Cadena de
                texto. Los posibles nombres de los tipos de dato
                binario se definen en
                geoVarGlobal.nombreDatoBinGeo{}
- dimDato:      Numero identificador de las dimensiones de los
                datos de altitud. Los posibles numeros
                indicadores de dimensiones se definen en
                geoVarGlobal.unidades{}
- tipoAlm:      Nombre del tipo de almacenamiento de los datos de
                altitud en el fichero binario correspondiente
                Los posibles nombres de los tipos de
                almacenamiento se definen en
                geoVarGlobal.tipoAlmacenBinarioGeo{}
- tipoDatoVar:  Nombre del tipo de dato binario de GEO utilizado
                para almacenar los datos de varianza. Cadena de
                texto
- dimDatoVar:   Numero identificador de las dimensiones de los
                datos de varianza
- tipoAlmVar:   Nombre del tipo de almacenamiento de los datos de
                varianza en el fichero binario correspondiente
```

```
                - noDato:      Indicador de existencia de puntos sin dato de
                               altitud. Dos posibilidades:
                               - 0: No
                               - 1: Si
                - idNoDato:    Valor identificativo de ausencia de dato de
                               altitud. Numero entero
                - noDatoVar:   Indicador de existencia de puntos sin dato de
                               varianza. Dos posibilidades:
                               - 0: No
                               - 1: Si
                - idNoDatoVar: Valor identificativo de ausencia de dato de
                               varianza. Numero entero
                - limNorte:    Limite norte de la malla de puntos (si el sistema
                               de coordenadas es geodesico, en radianes)
                - limSur:      Limite sur de la malla de puntos (si el sistema
                               de coordenadas es geodesico, en radianes)
                - limEste:     Limite este de la malla de puntos (si el sistema
                               de coordenadas es geodesico, en radianes)
                - limOeste:    Limite oeste de la malla de puntos (si el sistema
                               de coordenadas es geodesico, en radianes)
                - pasoLat:     Paso de malla en latitud (si el sistema de
                               coordenadas es geodesico, en radianes)
                - pasoLon:     Paso de malla en longitud (si el sistema de
                               coordenadas es geodesico, en radianes)
                - comentario:  Comentario acerca del MDE

    Ejemplo:    mde=geo_Mde('iberia30seg');

    Usa geoVarGlobal: Si

    Original:   04-05-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

    Revisiones: 09-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
                  Reorganizacion de la ayuda
                  Reescritura de la funcion para que no utilice la variable
                  varargin como contenedor de los argumentos de entrada
                  Adicion de sentencias de test
```

## 13.2. Carpeta 02-altitudMde

En esta carpeta se almacenan las funciones de interpolación de altitudes en los modelos digitales de elevaciones.

- **geo_AltitudMdeMalla.m**:

```
    Funcion:    geo_AltitudMdeMalla.m

    Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

    Proposito:  Calcula la altitud del terreno a partir de una malla de puntos de
                cota conocida y calcula una estimacion del error cometido
                (varianza)

    Entradas:   - Latitud geodesica (en radianes) o coordenada X del punto de
                  trabajo (escalar o vector columna)
                - Longitud geodesica (en radianes) o coordenada Y del punto de
                  trabajo (escalar o vector columna)
```

```
              - Tamano de la ventana de interpolacion, 1 o numero par entre 2 y
                10. Si el numero es 1 se utiliza el vecino mas proximo como
                solucion, si es 2 se realiza una interpolacion bilineal. En otro
                caso, la interpolacion se realiza por medio de splines cubicos
              - Estructura MDE
              Argumentos opcionales:
                - Indicador de calculo de error, numero entero. Si este
                  argumento toma el valor -1 o se omite no se realiza calculo
                  del error. En otro caso, el calculo se lleva a cabo. Si se
                  pide que calcule error y el MDE no dispone de informacion, el
                  calculo no se realiza y la funcion lanza un warning

  Salidas:    - Altitud del terreno (escalar o vector columna), en las mismas
                unidades que las de las altitudes dato de la malla de puntos
              - Varianza de la altitud calculada (escalar o vector columna), en
                las mismas unidades (al cuadrado) que las de la altitud
                En el caso de no haber pasado datos para el calculo del error
                este argumento de salida tiene el valor -1 (si se ha solicitado)

  Ejemplo:    [H]=geo_AltitudMdeMalla(latGeo,lonGeo,ventana,MDE);
              [H]=geo_AltitudMdeMalla(latGeo,lonGeo,ventana,MDE,-1);
              [H,varH]=geo_AltitudMdeMalla(latGeo,lonGeo,ventana,MDE,1);

  Funciones utilizadas de GEO: - geo_EsEscalar.m
                               - geo_EsMde.m
                               - geo_ControlDominioInterpMalla.m
                               - geo_DesactivaControles.m
                               - geo_ExtraeVentanaMalla.m
                               - geo_WarningOct.m
                               - geo_InterpolaMalla.m
                               - geo_QuitaEspacios.m
                               - geo_RestauraControles.m

  Funciones relacionadas de GEO: - geo_AltitudMdeTin.m
                                 - geo_OndGeoideMalla.m

  Ultima revision: 04-03-08

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_AltitudMdeTin.m**:

```
  Funcion:    geo_AltitudMdeTin.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Interpola altitudes en una triangulacion de Delaunay y calcula una
              estimacion del error cometido (varianza)

  Entradas:   - Vector (o matriz) de dos columnas que contiene las coordenadas
                en el plano de los puntos a interpolar:
                puntoInterp = [xp yp];
                Cada fila corresponde a un punto:
                - Col. 1: Coordenada X del punto a interpolar
                - Col. 2: Coordenada Y del punto a interpolar
              - Matriz de Nx3 elementos que almacena las coordenadas de los
                puntos de la nube triangulada. Cada fila corresponde a un
                punto:
```

```
                - Col. 1: Coordenada X
                - Col. 2: Coordenada Y
                - Col. 3: Coordenada Z
              - Matriz de Tx3 elementos que contiene la triangulacion de
                Delaunay de la nube de puntos pasada. Cada fila describe cada
                uno de los T triangulos que la forman. Cada columna es la
                posicion en la matriz de la nube de puntos que corresponde a
                cada vertice del triangulo
              - Numero natural que indica el numero de puntos de la nube (los
                mas cercanos) que han de ser tenidos en cuenta para realizar la
                interpolacion en el caso de que algun punto a interpolar quede
                fuera de la red de triangulos. En este caso se realizara una
                interpolacion ponderada con una potencia del inverso de la
                distancia a estos puntos mas cercanos
              - Numero real que indica la potencia a la que se elevara el
                inverso de la distancia para ponderar la media
              Argumentos opcionales:
                - Escalar o vector que contiene informacion sobre la varianza de
                  la coordenada Z de los puntos de la nube que forman la
                  triangulacion:
                  varCov=[SZ^2];
                Las varianzas han de ser pasadas en valor absoluto (no se
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envian cuatro
                variables de valor -1, no se hace el calculo del error

Salidas:      - Coordenada Z interpolada en las coordenadas pasadas (escalar o
                vector columna), en las mismas unidades que las de la coordenada
                Z de los puntos de la triangulacion
              - Varianza de la coordenada Z interpolada (escalar o vector
                columna), en las mismas unidades (al cuadrado) que las de la
                coordenada Z
                En el caso de no haber pasado datos para el calculo del error
                este argumento de salida tiene el valor -1 (si se ha solicitado)
              - Vector columna indicador de los puntos interpolados que estan
                dentro o fuera de la triangulacion. Tiene el mismo numero de
                filas que la matriz de puntos a interpolar y los valores
                posibles son:
                - 1: Si el punto correspondiente (con el mismo indice) en la
                     matriz esta contenido en la triangulacion. En este caso la
                     interpolacion se reliza en el triangulo que contiene al
                     punto
                - 0: Si el punto correspondiente (con el mismo indice) en la
                     matriz no esta contenido en la triangulacion. En este caso
                     la coordenada Z (y su varianza, si ha lugar) sera
                     interpolada ponderando con una potencia del inverso de la
                     distancia teniendo en cuenta el numero de puntos mas
                     cercanos de la nube de puntos de la triangulacion indicados
                     en el cuarto argumento de entrada de la funcion

Ejemplo:      [Z,varZ,dentro]=geo_AltitudMdeTin(interp,nube,tin,nPunt,pot);
              [Z,varZ,dentro]=geo_AltitudMdeTin(interp,nube,tin,nPunt,pot,-1);
              [Z,varZ,dentro]=geo_AltitudMdeTin(interp,nube,tin,nPunt,pot,
              varCovZ);

Funciones utilizadas de GEO: - geo_EsNatural.m
                             - geo_EsEscalar.m
```

```
                              - geo_HayQueCalcularErrores.m
                              - geo_Find.m
                              - geo_DesactivaControles.m
                              - geo_AjustaFilasVectorMatriz.m
                              - geo_InterpolaTin.m
                              - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_InterpolaInvDistEspacio.m
                               - geo_InterpolaInvDistPlano.m
                               - geo_InterpolaTin.m
                               - geo_InterpolaTriangulo.m

Ultima revision: 09-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

# Capítulo 14
# Sección 06-calculosGeodesicos

En esta carpeta se almacenan funciones relacionadas con el transporte de coordenadas.

## 14.1. Carpeta 01-dominiosGeodesicos

En esta carpeta se almacenan funciones relacionadas con el dominio de las coordenadas geodésicas.

- **geo_PonAcimutDominio.m**:

```
Funcion:    geo_PonAcimutDominio.m

Proposito:  Reduce un valor angular al dominio clasico del acimut: [0 2*pi[

Entradas:   - Valor angular (escalar o vector columna), en radianes

Salidas:    - Variable reducida al dominio [0 2*pi[ (escalar o vector columna)
              en radianes

Ejemplo:    acimut=geo_PonAcimutDominio(angulo);

Usa geoVarGlobal: Si

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_PonLatGeoDominio.m**:

```
Funcion:    geo_PonLatGeoDominio.m

Proposito:  Reduce un valor angular al dominio clasico de la latitud geodesica
            ]-pi/2 pi/2[

Entradas:   - Valor angular (escalar o vector columna), en radianes

Salidas:    - Variable reducida al dominio ]-pi/2 pi/2[ (escalar o vector
              columna), en radianes

Ejemplo:    latitudGeodesica=geo_PonLatGeoDominio(angulo);
```

```
Usa geoVarGlobal: Si

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_PonLonGeoDominio.m**:

```
Funcion:    geo_PonLonGeoDominio.m

Proposito:  Reduce un valor angular al dominio clasico de la longitud
            geodesica: ]-pi pi]

Entradas:   - Valor angular (escalar o vector columna), en radianes

Salidas:    - Variable reducida al dominio ]-pi pi] (escalar o vector columna)
              en radianes

Ejemplo:    longitudGeodesica=geo_PonLonGeoDominio(angulo);

Usa geoVarGlobal: Si

Original:   25-12-2005, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

## 14.2. Carpeta 02-transformacionSistCoord

En esta carpeta se guardan las funciones relacionadas con la transformación entre sistemas de coordenadas.

- **geo_EnuTri.m**:

```
Funcion:    geo_EnuTri.m

Proposito:  Pasa de un punto expresado en coordenadas cartesianas
            topocentricas (east, north, up) a un vector expresado en
            incrementos de coordenadas cartesianas tridimensionales
            geocentricas con origen en el origen del sistema topocentrico y
            calcula una estimacion del error cometido (matriz
            varianza-covarianza)
            La formulacion ha sido extraida de [GA-HL-GEO], pags. 59-60

Entradas:   - Latitud geodesica del origen del sistema topocentrico (escalar o
              vector columna), en radianes
            - Longitud geodesica del origen del sistema topocentrico (escalar
              o vector columna), en radianes
```

```
              - Coordenada X topocentrica del punto (escalar o vector columna)
              - Coordenada Y topocentrica del punto (escalar o vector columna)
              - Coordenada Z topocentrica del punto (escalar o vector columna)
              Argumentos opcionales:
                - Vector o matriz que contiene informacion sobre la matriz
                  varianza-covarianza de los puntos de trabajo. Dos
                  posibilidades:
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto. Queda asi
                    expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sx^2 Sy^2 Sz^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
                Si, en vez de las matrices varianza-covarianza, se envia una
                variable de valor -1, no se hace el calculo del error

Salidas:      - Incremento en la coordenada X en el sistema geocentrico del
                vector solucion (escalar o vector columna)
              - Incremento en la coordenada Y en el sistema geocentrico del
                vector solucion (escalar o vector columna)
              - Incremento en la coordenada Z en el sistema geocentrico del
                vector solucion (escalar o vector columna)
              - Matriz varianza-covarianza del vector obtenido mediante la
                transformacion
                Si en la entrada se ha utilizado una matriz diagonal, la salida
                tambien sera diagonal, de la forma:
                varCov=[Sdx^2 Sdy^2 Sdz^2];
                Si en la entrada se ha utilizado una matriz completa, la salida
                tambien sera completa, de la forma:
                varCov=[Sdx^2 Sdxdy Sdxdz Sdy^2 Sdydz Sdz^2];
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:      [dx,dy,dz]=geo_EnuTri(lat,lon,X,Y,Z);
              [dx,dy,dz]=geo_EnuTri(lat,lon,X,Y,Z,-1);
              [dx,dy,dz,varCov]=geo_EnuTri(lat,lon,X,Y,Z,varCov);

Usa geoVarGlobal: Si

Original:     07-06-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_GeoTri.m**:

```
Funcion:      geo_GeoTri.m

Proposito:  Pasa de coordenadas geodesicas a coordenadas cartesianas
            tridimensionales geocentricas y calcula una estimacion del error
            cometido (matriz varianza-covarianza)
            La formulacion ha sido extraida de [MPE-2001], pag. 111
```

```
Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Altitud elipsoidal del punto de trabajo (escalar o vector
              columna)
            - Semieje mayor del elipsoide, en las mismas unidades que la
              altitud elipsoidal
            - Aplanamiento del elipsoide, (a-b)/a
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto de trabajo. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Slat^2 Slon^2 Sh^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Slat^2 Slatlon Slath Slon^2 Slonh Sh^2];
              Si, en vez de las matrices varianza-covarianza, se envia una
              variable de valor -1, no se hace el calculo del error

Salidas:    - Coordenada X geocentrica (escalar o vector columna), en las
              mismas unidades que las de la altitud elipsoidal
            - Coordenada Y geocentrica (escalar o vector columna), en las
              mismas unidades que las de la altitud elipsoidal
            - Coordenada Z geocentrica (escalar o vector columna), en las
              mismas unidades que las de la altitud elipsoidal
            - Matriz varianza-covarianza del punto transformado, en las mismas
              unidades (al cuadrado para las varianzas) que las de la altitud
              elipsoidal
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sx^2 Sy^2 Sz^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [X,Y,Z]=geo_GeoTri(latGeo,lonGeo,hElip,a,f);
            [X,Y,Z]=geo_GeoTri(latGeo,lonGeo,hElip,a,f,-1);
            [X,Y,Z,varCov]=geo_GeoTri(latGeo,lonGeo,hElip,a,f,varCov);

Usa geoVarGlobal: Si

Original:   05-07-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_TriEnu.m**:

```
Funcion:    geo_TriEnu.m

Proposito:  Pasa de un vector expresado como incrementos de coordenadas
            cartesianas tridimensionales geocentricas a un punto expresado en
            coordenadas cartesianas topocentricas (east, north, up) con origen
            en el punto inicial del vector y calcula una estimacion del error
            cometido (matriz varianza-covarianza)
            La formulacion ha sido extraida de [GA-HL-GEO], pags. 59-60

Entradas:   - Latitud geodesica del punto origen del vector (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto origen del vector (escalar o vector
              columna), en radianes
            - Incremento en la coordenada X geocentrica del vector (escalar o
              vector columna)
            - Incremento en la coordenada Y geocentrica del vector (escalar o
              vector columna)
            - Incremento en la coordenada Z geocentrica del vector (escalar o
              vector columna)
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza de los vectores de trabajo. Dos
                posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada incremento de coordenadas del
                  vector. Queda asi expresada una matriz varianza-covarianza
                  diagonal:
                  varCov=[Sdx^2 Sdy^2 Sdz^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada incremento de coordenadas del
                  vector y sus correlaciones. Queda asi expresada (por filas)
                  una matriz varianza-covarianza completa:
                  varCov=[Sdx^2 Sdxdy Sdxdz Sdy^2 Sdydz Sdz^2];
              Si, en vez de las matrices varianza-covarianza, se envia una
              variable de valor -1, no se hace el calculo del error

Salidas:    - Coordenada X en el sistema topocentrico (escalar o vector
              columna)
            - Coordenada Y en el sistema topocentrico (escalar o vector
              columna)
            - Coordenada Z en el sistema topocentrico (escalar o vector
              columna)
            - Matriz varianza-covarianza del punto obtenido mediante la
              transformacion
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sx^2 Sy^2 Sz^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [x,y,z]=geo_TriEnu(lat,lon,dX,dY,dZ);
            [x,y,z]=geo_TriEnu(lat,lon,dX,dY,dZ,-1);
            [x,y,z,varCov]=geo_TriEnu(lat,lon,dX,dY,dZ,varCov);
```

```
Usa geoVarGlobal: Si

Nota:       No es necesario introducir la matriz de varianza-covarianza de las
            coordenadas del punto origen del vector porque se supone que este
            error ya esta contenido en la matriz de varianza-covarianza del
            vector mismo

Original:   07-06-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_TriGeo.m**:

```
Funcion:    geo_TriGeo.m

Proposito:  Pasa de coordenadas cartesianas tridimensionales geocentricas a
            coordenadas geodesicas y calcula una estimacion del error cometido
            (matriz varianza-covarianza)
            La formulacion ha sido extraida de [MPE-2001], pag. 112

Entradas:   - Coordenada X geocentrica (escalar o vector columna)
            - Coordenada Y geocentrica (escalar o vector columna)
            - Coordenada Z geocentrica (escalar o vector columna)
            - Semieje mayor del elipsoide, en las mismas unidades que las
              coordenadas geocentricas
            - Aplanamiento del elipsoide, (a-b)/a
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto de trabajo. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sx^2 Sy^2 Sz^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
              Si, en vez de las matrices varianza-covarianza, se envia una
              variable de valor -1, no se hace el calculo del error

Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Altitud elipsoidal del punto de trabajo (escalar o vector
              columna), en las mismas unidades que las coordenadas cartesianas
              geocentricas
            - Matriz varianza-covarianza del punto transformado (al cuadrado
              para las varianzas), en radianes la latitud y la longitud y en
              las mismas unidades que las coordenadas cartesianas geocentricas
              la altitud elipsoidal
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
```

```
              varCov=[Slat^2 Slon^2 Sh^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Slat^2 Slatlon Slath Slon^2 Slonh Sh^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [lat,lon,h]=geo_TriGeo(X,Y,Z,a,f);
            [lat,lon,h]=geo_TriGeo(X,Y,Z,a,f,-1);
            [lat,lon,h,varCov]=geo_TriGeo(X,Y,Z,a,f,varCov);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

## 14.3. Carpeta 03-latitudes

En esta carpeta se almacenan funciones relacionadas con la transformación entre las diferentes latitudes utilizadas en geodesia.

- **geo_LatAutLatGeo.m**:

```
Funcion:    geo_LatAutLatGeo.m

Proposito:  Pasa de latitud autalica a latitud geodesica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 16

Entradas:   - Latitud autalica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud autalica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slg^2, de la latitud geodesica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latGeo]=geo_LatAutLatGeo(latitudAutalica,a,f);
            [latGeo]=geo_LatAutLatGeo(latitudAutalica,a,f,-1);
            [latGeo,varLatGeo]=geo_LatAutLatGeo(latitudAutalica,a,f,
            varLatAut);

Usa geoVarGlobal: Si
```

```
Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatConfLatGeo.m**:

```
Funcion:    geo_LatConfLatGeo.m

Proposito:  Pasa de latitud conforme a latitud geodesica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 15

Entradas:   - Latitud conforme del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud conforme (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slg^2, de la latitud geodesica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latGeo]=geo_LatConfLatGeo(latitudConforme,a,f);
            [latGeo]=geo_LatConfLatGeo(latitudConforme,a,f,-1);
            [latGeo,varLatGeo]=geo_LatConfLatGeo(latitudConforme,a,f,
            varLatConf);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatGeoLatAut.m**:

```
Funcion:    geo_LatGeoLatAut.m

Proposito:  Pasa de latitud geodesica a latitud autalica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 16
```

```
Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud autalica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Sla^2, de la latitud autalica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latAut]=geo_LatGeoLatAut(latitudGeodesica,a,f);
            [latAut]=geo_LatGeoLatAut(latitudGeodesica,a,f,-1);
            [latAut,varLatAut]=geo_LatGeoLatAut(latitudGeodesica,a,f,
            varLatGeo);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatGeoLatConf.m**:

```
Funcion:    geo_LatGeoLatConf.m

Proposito:  Pasa de latitud geodesica a latitud conforme y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 15

Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud conforme del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slc^2, de la latitud conforme, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latConf]=geo_LatGeoLatConf(latitudGeodesica,a,f);
```

```
            [latConf]=geo_LatGeoLatConf(latitudGeodesica,a,f,-1);
            [latConf,varLatConf]=geo_LatGeoLatConf(latitudGeodesica,a,f,
            varLatGeo);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatGeoLatGeoc.m**:

```
Funcion:    geo_LatGeoLatGeoc.m

Proposito:  Pasa de latitud geodesica a latitud geocentrica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 17

Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud geocentrica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slg^2, de la latitud geocentrica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latGeoc]=geo_LatGeoLatGeoc(latitudGeodesica,a,f);
            [latGeoc]=geo_LatGeoLatGeoc(latitudGeodesica,a,f,-1);
            [latGeoc,varLatGeoc]=geo_LatGeoLatGeoc(latitudGeodesica,a,f,
            varLatGeo);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatGeoLatIso.m**:

```
Funcion:    geo_LatGeoLatIso.m
```

```
Proposito:  Pasa de latitud geodesica a latitud isometrica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 15

Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud isometrica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Sli^2, de la latitud isometrica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latIso]=geo_LatGeoLatIso(latitudGeodesica,a,f);
            [latIso]=geo_LatGeoLatIso(latitudGeodesica,a,f,-1);
            [latIso,varLatIso]=geo_LatGeoLatIso(latitudGeodesica,a,f,
            varLatGeo);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatGeoLatRect.m**:

```
Funcion:    geo_LatGeoLatRect.m

Proposito:  Pasa de latitud geodesica a latitud rectificante y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 17

Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud rectificante del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slr^2, de la latitud rectificante, en
```

```
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latRect]=geo_LatGeoLatRect(latitudGeodesica,a,f);
            [latRect]=geo_LatGeoLatRect(latitudGeodesica,a,f,-1);
            [latRect,varLatRect]=geo_LatGeoLatRect(latitudGeodesica,a,f,
            varLatGeo);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatGeoLatRed.m**:

```
Funcion:    geo_LatGeoLatRed.m

Proposito:  Pasa de latitud geodesica a latitud reducida y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 18

Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud reducida del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slr^2, de la latitud reducida, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latRed]=geo_LatGeoLatRed(latitudGeodesica,a,f);
            [latRed]=geo_LatGeoLatRed(latitudGeodesica,a,f,-1);
            [latRed,varLatRed]=geo_LatGeoLatRed(latitudGeodesica,a,f,
            varLatGeo);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatGeocLatGeo.m**:

```
Funcion:    geo_LatGeocLatGeo.m

Proposito:  Pasa de latitud geocentrica a latitud geodesica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 17

Entradas:   - Latitud geocentrica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geocentrica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slg^2, de la latitud geodesica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latGeo]=geo_LatGeocLatGeo(latitudGeocentrica,a,f);
            [latGeo]=geo_LatGeocLatGeo(latitudGeocentrica,a,f,-1);
            [latGeo,varLatGeo]=geo_LatGeocLatGeo(latitudGeocentrica,a,f,
            varLatGeoc);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatIsoLatGeo.m**:

```
Funcion:    geo_LatIsoLatGeo.m

Proposito:  Pasa de latitud isometrica a latitud geodesica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 15

Entradas:   - Latitud isometrica del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud isometrica (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error
```

```
Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slg^2, de la latitud geodesica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latGeo]=geo_LatIsoLatGeo(latitudIsometrica,a,f);
            [latGeo]=geo_LatIsoLatGeo(latitudIsometrica,a,f,-1);
            [latGeo,varLatGeo]=geo_LatIsoLatGeo(latitudIsometrica,a,f,
            varLatIso);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatRectLatGeo.m**:

```
Funcion:    geo_LatRectLatGeo.m

Proposito:  Pasa de latitud rectificante a latitud geodesica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 17

Entradas:   - Latitud rectificante del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud rectificante (escalar o vector
                columna), en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slg^2, de la latitud geodesica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latGeo]=geo_LatRectLatGeo(latitudRectificante,a,f);
            [latGeo]=geo_LatRectLatGeo(latitudRectificante,a,f,-1);
            [latGeo,varLatGeo]=geo_LatRectLatGeo(latitudRectificante,a,f,
            varLatRect);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

- **geo_LatRedLatGeo.m**:

```
Funcion:    geo_LatRedLatGeo.m

Proposito:  Pasa de latitud reducida a latitud geodesica y calcula una
            estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 18

Entradas:   - Latitud reducida del punto de trabajo (escalar o vector
              columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud reducida (escalar o vector columna),
                en radianes^2
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Estimacion de la varianza, Slg^2, de la latitud geodesica, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latGeo]=geo_LatRedLatGeo(latitudReducida,a,f);
            [latGeo]=geo_LatRedLatGeo(latitudReducida,a,f,-1);
            [latGeo,varLatGeo]=geo_LatRedLatGeo(latitudReducida,a,f,
            varLatRed);

Usa geoVarGlobal: Si

Original:   10-02-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-01-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Adicion de sentencias de test
```

## 14.4. Carpeta 04-calculosPlano

En esta carpeta se almacenan funciones relacionadas con el transporte de coordenadas sobre el plano.

- **geo_AcimutTopografico.m**:

```
Funcion:    geo_AcimutTopografico.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
Proposito:  Calcula el acimut sobre el plano entre dos puntos y una estimacion
            de su error (varianza)

Entradas:   - Coordenada X del punto inicial (escalar o vector columna)
            - Coordenada Y del punto inicial (escalar o vector columna)
            - Coordenada X del punto final (escalar o vector columna)
            - Coordenada Y del punto final (escalar o vector columna)
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto inicial. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Syi^2];
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto final. Identica estructura que
                en el punto anterior (han de coincidir las columnas de las dos
                matrices
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, no se hace el calculo del error

Salidas:    - Acimut topografico entre los puntos (escalar o vector columna),
              en radianes
            - Estimacion de la varianza, Saci^2, del acimut calculado, en
              radianes al cuadrado. En el caso de no haber pasado datos para
              el calculo del error, este argumento de salida tiene el valor -1
              (si se ha solicitado)

Ejemplo:    [aci]=geo_AcimutTopografico(Xi,Yi,Xf,Yf);
            [aci]=geo_AcimutTopografico(Xi,Yi,Xf,Yf,-1,-1);
            [aci,varAci]=geo_AcimutTopografico(Xi,Yi,Xf,Yf,varCovi,varCovf);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_Find.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_DistanciaEspacio.m
                               - geo_DistanciaPlano.m

Ultima revision: 11-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DistanciaPlano.m**:

```
Funcion:    geo_DistanciaPlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
Proposito:  Calcula la distancia entre dos puntos sobre el plano y una
            estimacion de su error (varianza)

Entradas:   - Coordenada X del punto inicial (escalar o vector columna)
            - Coordenada Y del punto inicial (escalar o vector columna)
            - Coordenada X del punto final (escalar o vector columna)
            - Coordenada Y del punto final (escalar o vector columna)
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto inicial. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Syi^2];
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto final. Identica estructura que
                en el punto anterior (han de coincidir las columnas de las dos
                matrices
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, no se hace el calculo del error

Salidas:    - Distancia entre los puntos (escalar o vector columna), en las
              mismas unidades que los datos de entrada
            - Estimacion de la varianza, Sd^2, de la distancia calculada, en
              las mismas unidades (al cuadrado) que los datos de entrada. En
              el caso de no haber pasado datos para el calculo del error, este
              argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [dist]=geo_DistanciaPlano(Xi,Yi,Xf,Yf);
            [dist]=geo_DistanciaPlano(Xi,Yi,Xf,Yf,-1,-1);
            [dist,varDist]=geo_DistanciaPlano(Xi,Yi,Xf,Yf,varCovi,varCovf);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_AcimutTopografico.m
                               - geo_DistanciaEspacio.m

Ultima revision: 09-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_IntDirectaPlano.m**:

```
Funcion:    geo_IntDirectaPlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve una interseccion directa en el plano y una estimacion del
```

```
            error cometido (varianza)
            La formulacion ha sido extraida de [OJ-MTOT-1984], pags. 116-117

Entradas:   - Matriz de observaciones, dos posibilidades:
              - De tamano 2x5, con la siguiente estructura:
                obs=[ N1  X1  Y1  HZX12  HZX1P
                      N2  X2  Y2  HZX21  HZX2P];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador de cada punto fijo
                - Col. 2: Coordenada X de cada punto fijo
                - Col. 3: Coordenada Y de cada punto fijo
                - Col. 4: Lectura horizontal del punto fijo estacion al otro
                          punto fijo, en radianes
                - Col. 5: Lectura horizontal del punto fijo estacion al punto
                          incognita, en radianes
              - De tamano 2x4, con la siguiente estructura:
                obs=[ X1  Y1  HZX12  HZX1P
                      X2  Y2  HZX21  HZX2P];
                Cada fila corresponde a una observacion:
                - Col. 1: Coordenada X de cada punto fijo
                - Col. 2: Coordenada Y de cada punto fijo
                - Col. 3: Lectura horizontal del punto fijo estacion al otro
                          punto fijo, en radianes
                - Col. 4: Lectura horizontal del punto fijo estacion al punto
                          incognita, en radianes
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de las lecturas horizontales. Dos
                posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada lectura horizontal. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Shzxij^2 Shzxip^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada lectura horizontal y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Shzxij^2 Shzxijhzxip Shzxip^2];
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos fijos. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Syi^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, no se hace el calculo del error

Salidas:    - Coordenada X del punto incognita, en las mismas unidades que las
```

```
                coordenadas de los puntos fijos
              - Coordenada Y del punto incognita, en las mismas unidades que las
                coordenadas de los puntos fijos
              - Matriz varianza-covarianza del punto incognita, en las mismas
                unidades (al cuadrado para las varianzas) que las de las
                coordenadas de los puntos fijos
                Si en la entrada se ha utilizado una matriz diagonal, la salida
                tambien sera diagonal, de la forma:
                varCov=[Sx^2 Sy^2];
                Si en la entrada se ha utilizado una matriz completa, la salida
                tambien sera completa, de la forma:
                varCov=[Sx^2 Sxy Sy^2];
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

  Ejemplo:    [X,Y]=geo_IntDirectaPlano(obs);
              [X,Y]=geo_IntDirectaPlano(obs,-1,-1);
              [X,Y,varCov]=geo_IntDirectaPlano(obs,varCovHz,varCovPtos);

  Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                               - geo_ControlDominio.m
                               - geo_VectorMatrizSimInterna.m

  Funciones relacionadas de GEO: - geo_IntDirectaPlanoH.m
                                 - geo_IntDosDistanciasPlano.m
                                 - geo_IntInversaPlano.m
                                 - geo_IntInversaPlanoH.m
                                 - geo_IntTresDistanciasPlano.m

  Ultima revision: 10-02-08

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_IntDirectaPlanoH.m**:

```
  Funcion:    geo_IntDirectaPlanoH.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Resuelve una interseccion directa en el plano teniendo en cuenta
              altimetria y una estimacion del error cometido (varianza)

  Entradas:   - Matriz de observaciones, dos posibilidades:
                - De tamano 2x9, con la siguiente estructura:
                  obs=[ N1  X1  Y1  Z1  HZX12  HZX1P  VX1P  HI1  HM1
                        N2  X2  Y2  Z2  HZX21  HZX2P  VX2P  HI2  HM2];
                  Cada fila corresponde a una observacion:
                  - Col. 1: Numero identificador de cada punto fijo
                  - Col. 2: Coordenada X de cada punto fijo
                  - Col. 3: Coordenada Y de cada punto fijo
                  - Col. 4: Coordenada Z de cada punto fijo
                  - Col. 5: Lectura horizontal del punto fijo estacion al otro
                            punto fijo, en radianes
                  - Col. 6: Lectura horizontal del punto fijo estacion al punto
                            incognita, en radianes
                  - Col. 7: Distancia cenital del punto fijo estacion al punto
                            incognita, en radianes
                  - Col. 8: Altura del instrumento de observacion, en las mismas
```

```
            unidades que las coordenadas de los puntos fijos
      - Col. 9: Altura del reflector/mira, en las mismas unidades
            que las coordenadas de los puntos fijos
    - De tamano 2x8, con la siguiente estructura:
      obs=[ X1  Y1  Z1  HZX12  HZX1P  VX1P  HI1  HM1
            X2  Y2  Z2  HZX21  HZX2P  VX2P  HI2  HM2];
      Cada fila corresponde a una observacion:
      - Col. 1: Coordenada X de cada punto fijo
      - Col. 2: Coordenada Y de cada punto fijo
      - Col. 3: Coordenada Z de cada punto fijo
      - Col. 4: Lectura horizontal del punto fijo estacion al otro
            punto fijo, en radianes
      - Col. 5: Lectura horizontal del punto fijo estacion al punto
            incognita, en radianes
      - Col. 6: Distancia cenital del punto fijo estacion al punto
            incognita, en radianes
      - Col. 7: Altura del instrumento de observacion, en las mismas
            unidades que las coordenadas de los puntos fijos
      - Col. 8: Altura del reflector/mira, en las mismas unidades
            que las coordenadas de los puntos fijos
    Los valores de todas las coordenadas han de estar en la misma
    magnitud lineal
  Argumentos opcionales:
    - Vector o matriz que contiene informacion sobre las matrices
      varianza-covarianza de las lecturas horizontales. Dos
      posibilidades:
      - Vector (o matriz) de dos columnas, cada una de las cuales
        contiene la varianza de cada lectura horizontal. Queda asi
        expresada una matriz varianza-covarianza diagonal:
        varCov=[Shzxij^2 Shzxip^2];
      - Vector (o matriz) de tres columnas, cada una de las cuales
        contiene la varianza de cada lectura horizontal y sus
        correlaciones. Queda asi expresada (por filas) una matriz
        varianza-covarianza completa:
        varCov=[Shzxij^2 Shzxijhzxip Shzxip^2];
    - Vector columna que contiene la varianza de cada observacion
      angular cenital
    - Vector columna que contiene la varianza de cada observacion
      de altura del instrumento
    - Vector columna que contiene la varianza de cada observacion
      de altura del reflector/mira
    - Vector o matriz que contiene informacion sobre las matrices
      varianza-covarianza de los puntos fijos. Dos posibilidades:
      - Vector (o matriz) de tres columnas, cada una de las cuales
        contiene la varianza de cada coordenada del punto. Queda asi
        expresada una matriz varianza-covarianza diagonal:
        varCov=[Sxi^2 Syi^2 Szi^2];
      - Vector (o matriz) de seis columnas, cada una de las cuales
        contiene la varianza de cada coordenada del punto y sus
        correlaciones. Queda asi expresada (por filas) una matriz
        varianza-covarianza completa:
        varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
    Las varianzas han de ser pasadas en valor absoluto (no se
    controla internamente)
    Si, en vez de las matrices varianza-covarianza, se envian
    variables de valor -1, no se hace el calculo del error
```

```
Salidas:    - Coordenada X del punto incognita, en las mismas unidades que las
              coordenadas de los puntos fijos
            - Coordenada Y del punto incognita, en las mismas unidades que las
              coordenadas de los puntos fijos
            - Coordenada Z del punto incognita, en las mismas unidades que las
              coordenadas de los puntos fijos. Resulta de una media entre las
              dos soluciones existentes. Si el problema se plantea con
              propagacion de errores la media resulta de una compensacion MMCC
              tomando como pesos las varianzas calculadas para cada altura. Si
              se plantea sin propagacion de errores el resultado es el fruto
              de una media ponderada con respecto a las distancias del punto
              solucion a los puntos estacion
            - Matriz varianza-covarianza del punto incognita, en las mismas
              unidades (al cuadrado para las varianzas) que las de las
              coordenadas de los puntos fijos
              Si en la entrada se han utilizado matrices diagonales, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sx^2 Sy^2 Sz^2];
              Si en la entrada se han utilizado matrices completas, la salida
              tambien sera completa, de la forma:
              varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [X,Y,Z]=geo_IntDirectaPlanoH(obs);
            [X,Y,Z]=geo_IntDirectaPlanoH(obs,-1,-1,-1,-1,-1);
            [X,Y,Z,varCov]=geo_IntDirectaPlanoH(obs,varCovHz,varCovV,varCovHI,
            varCovHM,varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_ControlDominio.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_IntDirectaPlano.m
                             - geo_RestauraControles.m
                             - geo_VectorMatrizSimInterna.m
                             - geo_MediaCoordenadas.m

Funciones relacionadas de GEO: - geo_IntDirectaPlano.m
                               - geo_IntDosDistanciasPlano.m
                               - geo_IntInversaPlano.m
                               - geo_IntInversaPlanoH.m
                               - geo_IntTresDistanciasPlano.m

Ultima revision: 10-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_IntDosDistanciasPlano.m**:

```
Funcion:    geo_IntDosDistanciasPlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve una interseccion de distancias en el plano y una
            estimacion del error cometido (varianza). Solo utiliza dos
            distancias (minimo imprescincdible), por lo que hay dos soluciones
            validas al problema
```

```
            Formulacion de Jose Luis Garcia Pallero

Entradas:   - Matriz de observaciones, dos posibilidades:
              - De tamano 2x4, con la siguiente estructura:
                obs=[ N1  X1  Y1  DIST1
                      N2  X2  Y2  DIST2];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador de cada punto fijo
                - Col. 2: Coordenada X de cada punto fijo
                - Col. 3: Coordenada Y de cada punto fijo
                - Col. 4: Distancia sobre el plano del punto fijo al punto
                          incognita
              - De tamano 2x3, con la siguiente estructura:
                obs=[ X1  Y1  DIST1
                      X2  Y2  DIST2];
                Cada fila corresponde a una observacion:
                - Col. 1: Coordenada X de cada punto fijo
                - Col. 2: Coordenada Y de cada punto fijo
                - Col. 3: Distancia sobre el plano del punto fijo al punto
                          incognita
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            Argumentos opcionales:
              - Vector columna que contiene la varianza de cada distancia
                observada
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los punto fijos. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Syi^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, no se hace el calculo del error

Salidas:    - Coordenadas X de los puntos solucion, vector columna de 2x1
              X=[ XP1
                  XP2];
              En las mismas unidades que los parametros de entrada
            - Coordenadas Y de los puntos solucion, vector columna de 2x1
              Y=[ YP1
                  YP2];
              En las mismas unidades que los parametros de entrada
              La primera fila de las matrices solucion corresponde al punto
              que, situados en el punto fijo numero 1 (primera fila de la
              matriz de observaciones) y mirando hacia el punto 2 (primera
              fila de la matriz de observaciones), queda a la derecha de la
              linea asi formada. La segunda fila de las matrices solucion
              corresponde al punto que queda a la izquierda
            - Matriz varianza-covarianza de los puntos incognita, en las
              mismas unidades (al cuadrado para las varianzas) que las de las
```

```
              coordenadas de los puntos fijos
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sxp1^2 Syp1^2
                      Sxp2^2 Syp2^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Sxp1^2 Sxp1yp1 Syp1^2
                      Sxp2^2 Sxp2yp2 Syp2^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [X,Y]=geo_IntDosDistanciasPlano(obs);
            [X,Y]=geo_IntDosDistanciasPlano(obs,-1,-1);
            [X,Y,varCov]=geo_IntDosDistanciasPlano(obs,varCovDist,varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_RestauraControles.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_IntDirectaPlano.m
                               - geo_IntDirectaPlanoH.m
                               - geo_IntInversaPlano.m
                               - geo_IntInversaPlanoH.m
                               - geo_IntTresDistanciasPlano.m

Ultima revision: 17-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_IntInversaPlano.m**:

```
Funcion:    geo_IntInversaPlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve una interseccion inversa en el plano y una estimacion del
            error cometido (varianza)
            La formulacion ha sido extraida de [OJ-MTOT-1984], pags. 153-156

Entradas:   - Matriz de observaciones, dos posibilidades:
              - De tamano 3x4, con la siguiente estructura:
                obs=[ N1  X1  Y1  LHZ1
                      N2  X2  Y2  LHZ2
                      N3  X3  Y3  LHZ3];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador de cada punto fijo
                - Col. 2: Coordenada X de cada punto fijo (observado)
                - Col. 3: Coordenada Y de cada punto fijo (observado)
                - Col. 3: Lectura angular horizontal del punto estacion al
                          punto fijo correspondiente a la fila, en radianes
              - De tamano 3x3, con la siguiente estructura:
                obs=[ X1  Y1  LHZ1
                      X2  Y2  LHZ2
                      X3  Y3  LHZ3];
                Cada fila corresponde a una observacion:
```

```
                - Col. 1: Coordenada X de cada punto fijo (observado)
                - Col. 2: Coordenada Y de cada punto fijo (observado)
                - Col. 3: Lectura angular horizontal del punto estacion al
                          punto fijo correspondiente a la fila, en radianes
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            Argumentos opcionales:
              - Vector columna que contiene la varianza de cada observacion
                angular
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los punto fijos. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Syi^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, no se hace el calculo del error

Salidas:    - Coordenada X del punto incognita, en las mismas unidades que las
              coordenadas de los puntos fijos
            - Coordenada Y del punto incognita, en las mismas unidades que las
              coordenadas de los puntos fijos
            - Matriz varianza-covarianza del punto incognita, en las mismas
              unidades (al cuadrado para las varianzas) que las de las
              coordenadas de los puntos fijos
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sx^2 Sy^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Sx^2 Sxy Sy^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [X,Y]=geo_IntInversaPlano(obs);
            [X,Y]=geo_IntInversaPlano(obs,-1,-1);
            [X,Y,varCov]=geo_IntInversaPlano(obs,varCovAng,varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_ControlDominio.m
                             - geo_DesactivaControles.m
                             - geo_OrdenaMatriz.m
                             - geo_AcimutTopografico.m
                             - geo_RestauraControles.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_IntDirectaPlano.m
                               - geo_IntDirectaPlanoH.m
                               - geo_IntDosDistanciasPlano.m
                               - geo_IntInversaPlanoH.m
```

```
                                - geo_IntTresDistanciasPlano.m

Ultima revision: 12-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_IntInversaPlanoH.m**:

```
Funcion:    geo_IntInversaPlanoH.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve una interseccion inversa en el plano teniendo en cuenta
            altimetria y una estimacion del error cometido (varianza)

Entradas:   - Matriz de observaciones, dos posibilidades:
              - De tamano 3x8, con la siguiente estructura:
                obs=[ N1  X1  Y1  Z1  LHZ1  V1  HI1  HM1
                      N2  X2  Y2  Z2  LHZ2  V2  HI2  HM2
                      N3  X3  Y3  Z3  LHZ3  V3  HI3  HM3];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador de cada punto fijo
                - Col. 2: Coordenada X de cada punto fijo (observado)
                - Col. 3: Coordenada Y de cada punto fijo (observado)
                - Col. 4: Coordenada Z de cada punto fijo (observado)
                - Col. 5: Lectura angular horizontal del punto estacion al
                          punto fijo correspondiente a la fila, en radianes
                - Col. 6: Distancia cenital del punto estacion al punto fijo
                          correspondiente a la fila, en radianes
                - Col. 7: Altura del instrumento de observacion, en las mismas
                          unidades que las coordenadas de los puntos fijos
                - Col. 8: Altura del reflector/mira, en las mismas unidades
                          que las coordenadas de los puntos fijos
              - De tamano 3x7, con la siguiente estructura:
                obs=[ X1  Y1  Z1  LHZ1  V1  HI1  HM1
                      X2  Y2  Z2  LHZ2  V2  HI2  HM2
                      X3  Y3  Z3  LHZ3  V3  HI3  HM3];
                Cada fila corresponde a una observacion:
                - Col. 1: Coordenada X de cada punto fijo (observado)
                - Col. 2: Coordenada Y de cada punto fijo (observado)
                - Col. 3: Coordenada Z de cada punto fijo (observado)
                - Col. 4: Lectura angular horizontal del punto estacion al
                          punto fijo correspondiente a la fila, en radianes
                - Col. 5: Distancia cenital del punto estacion al punto fijo
                          correspondiente a la fila, en radianes
                - Col. 6: Altura del instrumento de observacion, en las mismas
                          unidades que las coordenadas de los puntos fijos
                - Col. 7: Altura del reflector/mira, en las mismas unidades
                          que las coordenadas de los puntos fijos
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            Argumentos opcionales:
              - Vector columna que contiene la varianza de cada observacion
                angular horizontal
              - Vector columna que contiene la varianza de cada observacion
                angular cenital
              - Vector columna que contiene la varianza de cada observacion
                de altura del instrumento
```

```
                - Vector columna que contiene la varianza de cada observacion
                  de altura del reflector/mira
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza de los punto fijos. Dos posibilidades:
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto. Queda asi
                    expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sxi^2 Syi^2 Szi^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
                Las varianzas han de ser pasadas en valor absoluto (no se
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envian dos
                variables de valor -1, no se hace el calculo del error

Salidas:    - Coordenada X del punto incognita, en las mismas unidades que las
                coordenadas de los puntos fijos
            - Coordenada Y del punto incognita, en las mismas unidades que las
                coordenadas de los puntos fijos
            - Coordenada Z del punto incognita, en las mismas unidades que las
                coordenadas de los puntos fijos. Resulta de una media entre las
                tres soluciones existentes. Si el problema se plantea con
                propagacion de errores la media resulta de una compensacion MMCC
                tomando como pesos las varianzas calculadas para cada altura. Si
                se plantea sin propagacion de errores el resultado es el fruto
                de una media ponderada con respecto al inverso de la distancia
                del punto solucion a los puntos estacion
            - Matriz varianza-covarianza del punto incognita, en las mismas
                unidades (al cuadrado para las varianzas) que las de las
                coordenadas de los puntos fijos
                Si en la entrada se han utilizado matrices diagonales, la salida
                tambien sera diagonal, de la forma:
                varCov=[Sx^2 Sy^2 Sz^2];
                Si en la entrada se han utilizado matrices completas, la salida
                tambien sera completa, de la forma:
                varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [X,Y,Z]=geo_IntInversaPlanoH(obs);
            [X,Y,Z]=geo_IntInversaPlanoH(obs,-1,-1,-1,-1,-1);
            [X,Y,Z,varCov]=geo_IntInversaPlanoH(obs,varCovHz,varCovV,varCovHI,
            varCovHM,varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_ControlDominio.m
                             - geo_Find.m
                             - geo_DescativaControles.m
                             - geo_IntInversaPlano.m
                             - geo_RestauraControles.m
                             - geo_VectorMatrizSimInterna.m
                             - geo_MediaCoordenadas.m

Funciones relacionadas de GEO: - geo_IntDirectaPlano.m
```

```
                                - geo_IntDirectaPlanoH.m
                                - geo_IntDosDistanciasPlano.m
                                - geo_IntInversaPlano.m
                                - geo_IntTresDistanciasPlano.m

Ultima revision: 13-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_IntTresDistanciasPlano.m**:

```
Funcion:    geo_IntTresDistanciasPlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve una interseccion de tres distancias en el plano y una
            estimacion del error cometido (varianza). Resuelve por MMCC
            Esta funcion puede dar un resultado erroneo si los tres puntos
            fijos son colineales o se acercan a esta condicion

Entradas:   - Matriz de observaciones, dos posibilidades:
              - De tamano 3x4, con la siguiente estructura:
                obs=[ N1  X1  Y1  DIST1
                      N2  X2  Y2  DIST2
                      N3  X3  Y3  DIST3];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador de cada punto fijo
                - Col. 2: Coordenada X de cada punto fijo
                - Col. 3: Coordenada Y de cada punto fijo
                - Col. 4: Distancia sobre el plano del punto fijo al punto
                          incognita
              - De tamano 3x3, con la siguiente estructura:
                obs=[ X1  Y1  DIST1
                      X2  Y2  DIST2
                      X3  Y3  DIST3];
                Cada fila corresponde a una observacion:
                - Col. 1: Coordenada X de cada punto fijo
                - Col. 2: Coordenada Y de cada punto fijo
                - Col. 3: Distancia sobre el plano del punto fijo al punto
                          incognita
            Los valores de todas las coordenadas han de estar en la misma
            magnitud lineal
            Argumentos opcionales:
              - Vector columna que contiene la varianza de cada distancia
                observada
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los punto fijos. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Syi^2];
            Las varianzas han de ser pasadas en valor absoluto (no se
            controla internamente)
```

```
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, no se hace el calculo del error

Salidas:    - Coordenadas X del punto solucion, en las mismas unidades que
              los parametros de entrada
            - Coordenadas Y del punto solucion, en las mismas unidades que
              los parametros de entrada
            - Matriz varianza-covarianza de los puntos incognita, en las
              mismas unidades (al cuadrado para las varianzas) que las de las
              coordenadas de los puntos fijos
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sxp1^2 Syp1^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Sxp1^2 Sxp1yp1 Syp1^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [X,Y]=geo_IntTresDistanciasPlano(obs);
            [X,Y]=geo_IntTresDistanciasPlano(obs,-1,-1);
            [X,Y,varCov]=geo_IntTresDistanciasPlano(obs,varCovDist,
            varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_IntDirectaPlano.m
                               - geo_IntDirectaPlanoH.m
                               - geo_IntDosDistanciasPlano.m
                               - geo_IntInversaPlano.m
                               - geo_IntInversaPlanoH.m

Ultima revision: 17-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PerimetroPlano.m**:

```
Funcion:    geo_PerimetroPlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula el perimetro de una region limitada por un conjunto de
            puntos y una estimacion del error cometido (varianza)

Entradas:   - Matriz de puntos, dos posibilidades:
              - De tamano nx3, con la siguiente estructura:
                obs=[ N1  X1  Y1
                      N2  X2  Y2
                     ...........
                      Nn  Xn  Yn];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador de cada punto
                - Col. 2: Coordenada X del punto
                - Col. 3: Coordenada Y del punto
```

```
                - De tamano nx2, con la siguiente estructura:
                  obs=[ X1  Y1
                        X2  Y2
                       .......
                        Xn  Yn];
                  Cada fila corresponde a una observacion:
                  - Col. 1: Coordenada X del punto
                  - Col. 2: Coordenada Y del punto
                Los valores de todas las coordenadas han de estar en la misma
                magnitud lineal
                No es necesario repetir las coordenadas del punto inicial en la
                matriz de datos
              Argumentos opcionales:
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza de los puntos dato. Dos posibilidades:
                  - Vector (o matriz) de dos columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto. Queda asi
                    expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sxi^2 Syi^2];
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sxi^2 Sxiyi Syi^2];
                Las varianzas han de ser pasadas en valor absoluto (no se
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envia una
                variables de valor -1, no se hace el calculo del error

Salidas:      - Perimetro de la region encerrada por los puntos dato, en las
                mismas unidades que las coordenadas de los puntos iniciales
              - Varianza del perimetro calculado, en las mismas unidades (al
                cuadrado) que las de las coordenadas de los puntos dato
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:      [per]=geo_PerimetroPlano(puntos);
              [per]=geo_PerimetroPlano(puntos,-1);
              [per,var]=geo_PerimetroPlano(puntos,varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_DistanciaPlano.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_SuperficiePlano.m

Ultima revision: 20-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_RadiacionPlano.m**:

```
Funcion:    geo_RadiacionPlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

```
Proposito:  Calcula las coordenadas de un punto en el plano por radiacion  y
            una estimacion del error cometido (varianza)

Entradas:   - Acimut del punto fijo al punto incognita (escalar o vector
              columna), en radianes
            - Distancia reducida del punto fijo estacion al punto incognita
            - Coordenada X del punto fijo (escalar o vector columna), en las
              mismas unidades que la distancia
            - Coordenada Y del punto fijo (escalar o vector columna), en las
              mismas unidades que la distancia
            Argumentos opcionales:
              - Vector columna que contiene la varianza de cada observacion de
                acimut, en radianes^2
              - Vector columna que contiene la varianza de cada observacion de
                distancia
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los punto fijos. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Syi^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Coordenada X de los puntos incognita, en las mismas unidades que
              la distancia
            - Coordenada Y de los puntos incognita, en las mismas unidades que
              la distancia
            - Matriz varianza-covarianza de los puntos incognita, en las
              mismas unidades (al cuadrado para las varianzas) que las de la
              distancia
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sx^2 Sy^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Sx^2 Sxy Sy^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [Xv,Yv]=geo_RadiacionPlano(aci,dist,Xe,Ye);
            [Xv,Yv]=geo_RadiacionPlano(aci,dist,Xe,Ye,-1,-1,-1);
            [Xv,Yv,varCov]=geo_RadiacionPlano(aci,dist,Xe,Ye,varCovAci,
            varCovDr,varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_ControlDominio.m
                             - geo_Find.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_VectorMatrizSimInterna.m
```

```
Funciones relacionadas de GEO: - geo_RadiacionPlanoH.m

Ultima revision: 20-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_RadiacionPlanoH.m**:

```
Funcion:    geo_RadiacionPlanoH.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula las coordenadas de un punto por radiacion teniendo en
            cuenta altimetria y una estimacion del error cometido (varianza)

Entradas:   - Acimut del punto fijo al punto incognita (escalar o vector
              columna), en radianes
            - Distancia cenital del punto fijo al punto incognita (escalar o
              vector columna), en radianes
            - Distancia geometrica del punto fijo estacion al punto incognita
            - Altura del instrumento de observacion (escalar o vector
              columna), en las mismas unidades que la distancia
            - Altura del reflector/mira (escalar o vector columna), en las
              mismas unidades que la distancia
            - Coordenada X del punto fijo (escalar o vector columna), en las
              mismas unidades que la distancia
            - Coordenada Y del punto fijo (escalar o vector columna), en las
              mismas unidades que la distancia
            - Coordenada Z del punto fijo (escalar o vector columna), en las
              mismas unidades que la distancia
            Argumentos opcionales:
              - Vector columna que contiene la varianza de cada observacion de
                acimut, en radianes^2
              - Vector columna que contiene la varianza de cada observacion
                angular cenital, en radianes^2
              - Vector columna que contiene la varianza de cada observacion de
                distancia
              - Vector columna que contiene la varianza de cada observacion de
                altura del instrumento
              - Vector columna que contiene la varianza de cada observacion de
                altura del reflector/mira
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos fijos. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2 Szi^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
            Las varianzas han de ser pasadas en valor absoluto (no se
            controla internamente)
            Si, en vez de las matrices varianza-covarianza, se envian
            variables de valor -1, no se hace el calculo del error
```

```
Salidas:    - Coordenada X de los puntos incognita, en las mismas unidades que
              la distancia
            - Coordenada Y de los puntos incognita, en las mismas unidades que
              la distancia
            - Coordenada Z de los puntos incognita, en las mismas unidades que
              la distancia
            - Matriz varianza-covarianza de los puntos incognita, en las
              mismas unidades (al cuadrado para las varianzas) que las de la
              distancia
              Si en la entrada se han utilizado matrices diagonales, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sx^2 Sy^2 Sz^2];
              Si en la entrada se han utilizado matrices completas, la salida
              tambien sera completa, de la forma:
              varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [Xv,Yv,Zv]=geo_RadiacionPlanoH(aci,v,dist,hi,hr,xe,ye,ze);
            [Xv,Yv,Zv]=geo_RadiacionPlanoH(aci,v,dist,hi,hr,xe,ye,ze,-1,-1,-1,
            -1,-1,-1);
            [Xv,Yv,Zv,varCov]=geo_RadiacionPlanoH(aci,v,dist,hi,hr,xe,ye,ze,
            varCovHz,varCovV,varCovDg,varCovHI,varCovHM,varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_ControlDominio.m
                             - geo_Find.m
                             - geo_HayQueCalcularErrores.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_RadiacionPlano.m

Ultima revision: 20-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_SuperficiePlano.m**:

```
Funcion:    geo_SuperficiePlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la superficie de una region limitada por un conjunto de
            puntos y una estimacion del error cometido (varianza)
            Las expresiones para las derivadas parciales de la formula que da
            la superficie han sido tomadas de [NG-TC-2004], pag 7. Ha sido
            modificado un signo debido a un error en la transcripcion de la
            publicacion. La formulacion correcta para la derivada parcial de
            la funcion que da la superficie con respecto a la coordenada Y
            para puntos que no sean ni el primero ni el ultimo de la lista es:
            dS/dyn=0.5*(X_(n+1)-X_(n-1)). En realidad, este error no repercute
            en el resultado final porque esta derivada parcial sera elevada al
            cuadrado y el cambio de signo quedara suprimido

Entradas:   - Matriz de puntos, dos posibilidades:
              - De tamano nx3, con la siguiente estructura:
                obs=[ N1  X1  Y1
```

```
                    N2  X2  Y2
                   ...........
                    Nn  Xn  Yn];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador de cada punto
                - Col. 2: Coordenada X del punto
                - Col. 3: Coordenada Y del punto
              - De tamano nx2, con la siguiente estructura:
                obs=[ X1  Y1
                      X2  Y2
                     .......
                      Xn  Yn];
                Cada fila corresponde a una observacion:
                - Col. 1: Coordenada X del punto
                - Col. 2: Coordenada Y del punto
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
              No es necesario repetir las coordenadas del punto inicial en la
              matriz de datos
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos dato. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Syi^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envia una
              variables de valor -1, no se hace el calculo del error

Salidas:    - Superficie de la region encerrada por los puntos dato, en las
              mismas unidades que las coordenadas de los puntos del perimetro
            - Varianza de la superficie calculada, en las mismas unidades (al
              cuadrado) que las de las coordenadas de los puntos dato
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [sup]=geo_SuperficiePlano(puntos);
            [sup]=geo_SuperficiePlano(puntos,-1);
            [sup,var]=geo_SuperficiePlano(puntos,varCovPtos);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_PerimetroPlano.m

Ultima revision: 21-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 14.5. Carpeta 05-calculosEspacio

En esta carpeta se almacenan funciones relacionadas con el transporte de coordenadas en el espacio.

- **geo_DistanciaCenital.m**:

```
Funcion:    geo_DistanciaCenital.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la distancia cenital entre dos puntos y una estimacion del
            error cometido (varianza)

Entradas:   - Altitud del punto estacion (escalar o vector columna)
            - Altitud del punto visado (escalar o vector columna)
            - Distancia geometrica del punto estacion al punto visado (escalar
              o vector columna)
            Todos los datos han de estar en la misma magnitud lineal
            Argumentos opcionales:
              - Vector columna que contiene la varianza de cada observacion de
                altitud del punto estacion
              - Vector columna que contiene la varianza de cada observacion de
                altitud del punto visado
              - Vector columna que contiene la varianza de cada observacion de
                distancia
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian
              variables de valor -1, no se hace el calculo del error

Salidas:    - Distancia cenital (escalar o vector columna) entre los puntos
              pasados, en radianes
            - Varianza de la distancia cenital calculada, en radiane^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [V]=geo_DistanciaCenital(Ze,Zv,Dg);
            [V]=geo_DistanciaCenital(Ze,Zv,Dg,-1,-1,-1);
            [V,varV]=geo_DistanciaCenital(Ze,Zv,Dg,varZe,varZv,varDg);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_Find.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 09-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_DistanciaEspacio.m**:

```
Funcion:    geo_DistanciaEspacio.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la distancia entre dos puntos en el espacio euclideo
            tridimensional y una estimacion de su error (varianza)
```

```
Entradas:   - Coordenada X del punto inicial (escalar o vector columna)
            - Coordenada Y del punto inicial (escalar o vector columna)
            - Coordenada Z del punto inicial (escalar o vector columna)
            - Coordenada X del punto final (escalar o vector columna)
            - Coordenada Y del punto final (escalar o vector columna)
            - Coordenada Z del punto final (escalar o vector columna)
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto inicial. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2 Szi^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto final. Identica estructura que
                en el punto anterior (han de coincidir las columnas de las dos
                matrices
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, no se hace el calculo del error

Salidas:    - Distancia entre los puntos (escalar o vector columna), en las
              mismas unidades que los datos de entrada
            - Estimacion de la varianza, Sd^2, de la distancia calculada, en
              las mismas unidades (al cuadrado) que los datos de entrada. En
              el caso de no haber pasado datos para el calculo del error, este
              argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [dist]=geo_DistanciaEspacio(Xi,Yi,Zi,Xf,Yf,Zf);
            [dist]=geo_DistanciaEspacio(Xi,Yi,Zi,Xf,Yf,Zf,-1,-1);
            [dist,varDist]=geo_DistanciaEspacio(Xi,Yi,Zi,Xf,Yf,Zf,varCovi,
            varCovf);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_AcimutTopografico.m
                               - geo_DistanciaPlano.m

Ultima revision: 09-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 14.6. Carpeta 06-calculosEsfera

En esta carpeta se almacenan funciones relacionadas con el transporte de coordenadas sobre la esfera.

- **geo_ProbDirGeoEsf.m**:

```
Funcion:    geo_ProbDirGeoEsf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve el problema directo de la geodesia sobre la esfera por el
            metodo del cambio de base y calcula una estimacion del error
            cometido (varianza)
            La formulacion ha sido extraida de [GA-HL-GEO], pags. 206-207

Entradas:   - Latitud geodesica del punto inicial (escalar o vector columna),
              en radianes
            - Longitud geodesica del punto inicial (escalar o vector columna),
              en radianes
            - Acimut de la linea geodesica en el punto de partida (escalar o
              vector columna), en radianes
            - Longitud de la linea geodesica (escalar o vector columna)
            - Radio de la esfera, en las mismas unidades que la longitud de la
              linea geodesica
            Argumentos opcionales:
              - Escalar o vector columna que contiene la varianza de cada
                observacion de acimut, en radianes^2
              - Escalar o vector columna que contiene la varianza de cada
                observacion de distancia
              - Vector que contiene informacion sobre las matrices
                varianza-covarianza del punto de partida. Dos posibilidades:
                - Vector o matriz de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada (por filas) una matriz varianza-covarianza
                  diagonal:
                  varCov=[Slat^2 Slon^2];
                - Vector o matriz de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Slat^2 Slatlon Slon^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Latitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Acimut reciproco de la linea geodesica (escalar o vector
              columna), en radianes
            - Matrices varianza-covarianza de los puntos de llegada
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma (por filas):
              varCov=[Slat^2 Slon^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma (por filas):
              varCov=[Slat^2 Slatlon Slon^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)
            - Varianza del acimut reciproco de la linea geodesica (escalar o
```

```
                vector columna)
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [lat2,lon2,aci21]=geo_ProbDirGeoEsf(lat1,lon1,aci12,dist12,radio);
            [lat2,lon2,aci21]=geo_ProbDirGeoEsf(lat1,lon1,aci12,dist12,radio,
            -1,-1,-1);
            [lat2,lon2,aci21,varCovPto2,varAci21]=geo_ProbDirGeoEsf(lat1,lon1,
            aci12,dist12,radio,varAci12,varDist12,varCovPto1);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoLonGeo.m
                             - geo_ControlDominio.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_WarningOct.m
                             - geo_VectorMatrizSimInterna.m
                             - geo_Find.m
                             - geo_PonAcimutDominio.m
                             - geo_PonLatGeoDominio.m
                             - geo_PonLonGeoDominio.m

Funciones relacionadas de GEO: - geo_ProbDirGeo.m
                               - geo_ProbDirGeoElip.m
                               - geo_ProbInvGeo.m
                               - geo_ProbInvGeoElip.m
                               - geo_ProbInvGeoEsf.m

Nota: Esta funcion puede dar un resultado inconsistente en lineas en las que
      el punto de llegada tenga (simultaneamente) una latitud cercana a los
      polos (>=abs(+-89.5 grados)) y una longitud de 180 grados

Ultima revision: 20-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbInvGeoEsf.m**:

```
Funcion:    geo_ProbInvGeoEsf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve el problema inverso de la geodesia sobre la esfera por el
            metodo del cambio de base y calcula una estimacion del error
            cometido (varianza)
            La formulacion ha sido extraida de [GA-HL-GEO], pags. 204-205

Entradas:   - Latitud geodesica del punto de partida (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de partida (escalar o vector
              columna), en radianes
            - Latitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Radio de la esfera
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre las matrices
```

```
                varianza-covarianza del punto de partida. Dos posibilidades:
                - Vector o matriz de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada (por filas) una matriz varianza-covarianza
                  diagonal:
                  varCov=[Slat^2 Slon^2];
                - Vector o matriz de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Slat^2 Slatlon Slon^2];
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza del punto de llegada. Las mismas
                posibilidades que en el caso de las matrices de los puntos de
                partida
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Longitud de la linea geodesica, en las mismas unidades que el
              radio de la esfera
            - Acimut directo (del punto de partida al de llegada) de la linea
              geodesica (escalar o vector columna), en radianes
            - Acimut reciproco (del punto de llegada al de partida) de la
              linea geodesica (escalar o vector columna), en radianes
            - Varianza de la longitud de la linea geodesica (escalar o vector
              columna)
            - Varianza del acimut directo de la linea geodesica (escalar o
              vector columna), en radianes^2
            - Varianza del acimut reciproco de la linea geodesica (escalar o
              vector columna), en radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              los argumentos de salida referentes a las varianzas tienen el
              valor -1 (si se han solicitado)

Ejemplo:    [dist12,aci12,aci21]=geo_ProbInvGeoEsf(lat1,lon1,lat2,lon2,radio);
            [dist12,aci12,aci21]=geo_ProbInvGeoEsf(lat1,lon1,lat2,lon2,radio,
            -1,-1,-1);
            [dist12,aci12,aci21,varDist,varAci12,varAci21]=geo_ProbInvGeoEsf(
            lat1,lon1,lat2,lon2,radio,varCovPto1,varCovPto2);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoLonGeo.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_WarningOct.m
                             - geo_VectorMatrizSimInterna.m
                             - geo_PonAcimutDominio.m

Funciones relacionadas de GEO: - geo_ProbDirGeo.m
                               - geo_ProbDirGeoElip.m
                               - geo_ProbDirGeoEsf.m
                               - geo_ProbInvGeo.m
                               - geo_ProbInvGeoElip.m

Nota: Esta funcion da resultados inconsistentes en el calculo del error de los
```

```
      acimutes directo y reciproco. Puede ser debido a inestabilidades en el
      calculo de las derivadas numericas

Ultima revision: 20-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 14.7. Carpeta 07-calculosElipsoide

En esta carpeta se almacenan funciones relacionadas con el transporte de coordenadas sobre el elipsoide.

- **geo_DistMerLatGeo.m**:

```
Funcion:    geo_DistMerLatGeo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la latitud geodesica correspondiente a una longitud de
            arco de meridiano desde el ecuador y una estimacion de su error
            (varianza)
            Formulacion de Jose Luis Garcia Pallero

Entradas:   - Longitud de un arco de meridiano desde el ecuador (escalar o
              vector columna)
              Cuando un punto esta en el hemisferio sur, la longitud del arco
              de meridiano correspondiente se expresa como un numero negativo
            - Semieje mayor del elipsoide, en las mismas unidades que la
              longitud del arco de meridiano
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la longitud del arco de meridiano (escalar o
                vector columna), en las mismas unidades (al cuadrado) que la
                longitud del arco de meridiano pasada
              La varianza ha de ser pasada en valor absoluto (no se controla
              internamente)
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Latitud geodesica correspondiente a la longitud del arco de
              meridiano (escalar o vector columna), en radianes
            - Estimacion de la varianza, Slg^2, de la latitud calculada, en
              radianes^2
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [latGeo]=geo_DistMerLatGeo(lam,a,f);
            [latGeo]=geo_DistMerLatGeo(lam,a,f,-1);
            [latGeo,varLatGeo]=geo_DistMerLatGeo(lam,a,f,varLam);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioAF.m
                             - geo_ControlDominio.m

Funciones relacionadas de GEO: - geo_LatGeoDistMer.m
```

```
Ultima revision: 10-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_LatGeoDistMer.m**:

```
Funcion:    geo_LatGeoDistMer.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la longitud de un arco de meridiano desde el ecuador hasta
            una latitud geodesica dada y una estimacion de su error (varianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pag. 15

Entradas:   - Latitud geodesica (escalar o vector columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              La varianza ha de ser pasada en valor absoluto (no se controla
              internamente)
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Longitud del arco de meridiano desde el ecuador hasta el punto
              de trabajo (escalar o vector columna), en las mismas unidades
              que las del semieje mayor pasado
              Cuando un punto esta en el hemisferio sur, la longitud del arco
              de meridiano correspondiente se expresa como un numero negativo
            - Estimacion de la varianza, Sl^2, de la longitud calculada, en
              las mismas unidades (al cuadrado) que las del semieje mayor
              pasado
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [lam]=geo_LatGeoDistMer(latitudGeodesica,a,f);
            [lam]=geo_LatGeoDistMer(latitudGeodesica,a,f,-1);
            [lam,varLam]=geo_LatGeoDistMer(latitudGeodesica,a,f,varLatGeo);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoAF.m

Funciones relacionadas de GEO: - geo_DistMerLatGeo.m

Ultima revision: 10-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbDirGeo.m**:

```
Funcion:    geo_ProbDirGeo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve el problema directo de la geodesia y calcula una
```

```
            estimacion del error cometido (varianza)

Entradas:   - Latitud geodesica del punto inicial (escalar o vector columna),
              en radianes
            - Longitud geodesica del punto inicial (escalar o vector columna),
              en radianes
            - Acimut de la linea geodesica en el punto de partida (escalar o
              vector columna), en radianes
            - Longitud de la linea geodesica (escalar o vector columna)
            - Semieje mayor del elipsoide, en las mismas unidades que la
              longitud de la geodesica
            - Aplanamiento del elipsoide, (a-b)/a. Si la superficie de trabajo
              es la esfera este argumento ha de valer 0
            Argumentos opcionales:
              - Escalar o vector columna que contiene la varianza de cada
                observacion de acimut, en radianes^2
              - Escalar o vector columna que contiene la varianza de cada
                observacion de distancia
              - Vector que contiene informacion sobre las matrices
                varianza-covarianza del punto de partida. Dos posibilidades:
                - Vector o matriz de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada (por filas) una matriz varianza-covarianza
                  diagonal:
                  varCov=[Slat^2 Slon^2];
                - Vector o matriz de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Slat^2 Slatlon Slon^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Latitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de llegada (escalar o vector
              columna), en radianes
            - Acimut reciproco de la linea geodesica (escalar o vector
              columna), en radianes
            - Matrices varianza-covarianza de los puntos de llegada
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma (por filas):
              varCov=[Slat^2 Slon^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma (por filas):
              varCov=[Slat^2 Slatlon Slon^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)
            - Varianza del acimut reciproco de la linea geodesica (escalar o
              vector columna)
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [lat2,lon2,aci21]=geo_ProbDirGeo(lat1,lon1,aci12,dist12,radio);
            [lat2,lon2,aci21]=geo_ProbDirGeo(lat1,lon1,aci12,dist12,radio,-1,
```

```
            -1,-1);
            [lat2,lon2,aci21,varCovPto2,varAci21]=geo_ProbDirGeo(lat1,lon1,
            aci12,dist12,radio,varAci12,varDist12,varCovPto1);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_ControlDominioLatGeoLonGeo.m
                             - geo_ControlDominio.m
                             - geo_ControlDominioAF.m
                             - geo_DesactivaControles.m
                             - geo_ProbDirGeoElip.m
                             - geo_ProbDirGeoEsf.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_ProbDirGeoElip.m
                               - geo_ProbDirGeoEsf.m
                               - geo_ProbInvGeo.m
                               - geo_ProbInvGeoElip.m
                               - geo_ProbInvGeoEsf.m

Nota 1: Esta funcion puede presentar problemas numericos de dominio cuando
        la longitud del punto de llegada es de 180 grados exactos, pudiendose
        presentar el resultado como -180
Nota 2: El calculo de la propagacion de errores mediante calculo numerico de
        las derivadas da problemas en esta funcion

Ultima revision: 10-09-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbDirGeoElip.m**:

```
Funcion:    geo_ProbDirGeoElip.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Resuelve el problema directo de la geodesia sobre el elipsoide por
            el metodo de integracion numerica de Runge-Kutta de 4o orden y
            calcula una estimacion del error cometido (varianza)
            La formulacion ha sido extraida de [GA-HL-GEO], pags. 211-215

Entradas:   - Latitud geodesica del punto inicial (escalar o vector columna),
              en radianes
            - Longitud geodesica del punto inicial (escalar o vector columna),
              en radianes
            - Acimut de la linea geodesica en el punto de partida (escalar o
              vector columna), en radianes
            - Longitud de la linea geodesica (escalar o vector columna)
            - Semieje mayor del elipsoide, en las mismas unidades que la
              longitud de la linea geodesica
            - Aplanamiento del elipsoide, (a-b)/a
            Argumentos opcionales:
              - Escalar o vector columna que contiene la varianza de cada
                observacion de acimut, en radianes^2
              - Escalar o vector columna que contiene la varianza de cada
                observacion de distancia
              - Vector que contiene informacion sobre las matrices
                varianza-covarianza del punto de partida. Dos posibilidades:
                - Vector o matriz de dos columnas, cada una de las cuales
```

```
                    contiene la varianza de cada coordenada del punto. Queda asi
                    expresada (por filas) una matriz varianza-covarianza
                    diagonal:
                    varCov=[Slat^2 Slon^2];
                  - Vector o matriz de tres columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Slat^2 Slatlon Slon^2];
                Las varianzas han de ser pasadas en valor absoluto (no se
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envian tres
                variables de valor -1, no se hace el calculo del error

Salidas:      - Latitud geodesica del punto de llegada (escalar o vector
                columna), en radianes
              - Longitud geodesica del punto de llegada (escalar o vector
                columna), en radianes
              - Acimut reciproco de la linea geodesica (escalar o vector
                columna), en radianes
              - Matrices varianza-covarianza de los puntos de llegada
                Si en la entrada se ha utilizado una matriz diagonal, la salida
                tambien sera diagonal, de la forma (por filas):
                varCov=[Slat^2 Slon^2];
                Si en la entrada se ha utilizado una matriz completa, la salida
                tambien sera completa, de la forma (por filas):
                varCov=[Slat^2 Slatlon Slon^2];
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)
              - Varianza del acimut reciproco de la linea geodesica (escalar o
                vector columna)
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:      [lat2,lon2,aci21]=geo_ProbDirGeoElip(lat1,lon1,aci12,dist12,a,f);
              [lat2,lon2,aci21]=geo_ProbDirGeoElip(lat1,lon1,aci12,dist12,a,f,
              -1,-1,-1);
              [lat2,lon2,aci21,varCovPto2,varAci21]=geo_ProbDirGeoElip(lat1,
              lon1,aci12,dist12,a,f,varAci12,varDist12,varCovPto1);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoLonGeo.m
                             - geo_ControlDominio.m
                             - geo_ControlDominioAF.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_WarningOct.m
                             - geo_RestauraCOntroles.m
                             - geo_VectorMatrizSimInterna.m
                             - geo_RadioElipMer.m
                             - geo_RadioPrimVert.m
                             - geo_PonLatGeoDominio.m
                             - geo_PonLonGeoDominio.m
                             - geo_PonAcimutDominio.m

Funciones relacionadas de GEO: - geo_ProbDirGeo.m
```

```
                                - geo_ProbDirGeoElip.m
                                - geo_ProbInvGeo.m
                                - geo_ProbInvGeoElip.m
                                - geo_ProbInvGeoEsf.m

   Nota: Esta funcion puede dar un resultado inconsistente en lineas en las que
         el punto de llegada tenga (simultaneamente) una latitud cercana a los
         polos (>=abs(+-89.5 grados)) y una longitud de 180 grados

   Ultima revision: 20-02-08

   Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbInvGeo.m**:

```
   Funcion:    geo_ProbInvGeo.m

   Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

   Proposito:  Resuelve el problema inverso de la geodesia y calcula una
               estimacion del error cometido (varianza)

   Entradas:   - Latitud geodesica del punto de partida (escalar o vector
                 columna), en radianes
               - Longitud geodesica del punto de partida (escalar o vector
                 columna), en radianes
               - Latitud geodesica del punto de llegada (escalar o vector
                 columna), en radianes
               - Longitud geodesica del punto de llegada (escalar o vector
                 columna), en radianes
               - Semieje mayor del elipsoide
               - Aplanamiento del elipsoide, (a-b)/a. Si la superficie de trabajo
                 es la esfera este argumento ha de valer 0
               Argumentos opcionales:
                 - Vector o matriz que contiene informacion sobre las matrices
                   varianza-covarianza del punto de partida. Dos posibilidades:
                   - Vector o matriz de dos columnas, cada una de las cuales
                     contiene la varianza de cada coordenada del punto. Queda asi
                     expresada (por filas) una matriz varianza-covarianza
                     diagonal:
                     varCov=[Slat^2 Slon^2];
                   - Vector o matriz de tres columnas, cada una de las cuales
                     contiene la varianza de cada coordenada del punto y sus
                     correlaciones. Queda asi expresada (por filas) una matriz
                     varianza-covarianza completa:
                     varCov=[Slat^2 Slatlon Slon^2];
                 - Vector o matriz que contiene informacion sobre las matrices
                   varianza-covarianza del punto de llegada. Las mismas
                   posibilidades que en el caso de las matrices de los puntos de
                   partida
                 Las varianzas han de ser pasadas en valor absoluto (no se
                 controla internamente)
                 Si, en vez de las matrices varianza-covarianza, se envian tres
                 variables de valor -1, no se hace el calculo del error

   Salidas:    - Longitud de la linea geodesica, en las mismas unidades que el
                 semieje mayor
               - Acimut directo (del punto de partida al de llegada) de la linea
```

```
                geodesica (escalar o vector columna), en radianes
              - Acimut reciproco (del punto de llegada al de partida) de la
                linea geodesica (escalar o vector columna), en radianes
              - Varianza de la longitud de la linea geodesica (escalar o vector
                columna)
              - Varianza del acimut directo de la linea geodesica (escalar o
                vector columna), en radianes^2
              - Varianza del acimut reciproco de la linea geodesica (escalar o
                vector columna), en radianes^2
                En el caso de no haber pasado datos para el calculo del error,
                los argumentos de salida referentes a las varianzas tienen el
                valor -1 (si se han solicitado)

  Ejemplo:    [dist12,aci12,aci21]=geo_ProbInvGeo(lat1,lon1,lat2,lon2,radio);
              [dist12,aci12,aci21]=geo_ProbInvGeo(lat1,lon1,lat2,lon2,radio,-1,
              -1,-1);
              [dist12,aci12,aci21,varDist,varAci12,varAci21]=geo_ProbInvGeo(
              lat1,lon1,lat2,lon2,radio,varCovPto1,varCovPto2);

  Funciones utilizadas de GEO: - geo_EsEscalar.m
                               - geo_ControlDominioLatGeoLonGeo.m
                               - geo_ControlDominioAF.m
                               - geo_DesactivaControles.m
                               - geo_ProbInvGeoElip.m
                               - geo_ProbInvGeoEsf.m
                               - geo_RestauraControles.m

  Funciones relacionadas de GEO: - geo_ProbDirGeo.m
                                 - geo_ProbDirGeoElip.m
                                 - geo_ProbDirGeoEsf.m
                                 - geo_ProbInvGeoElip.m
                                 - geo_ProbInvGeoEsf.m

  Nota: El calculo de la propagacion de errores mediante calculo numerico de las
        derivadas da problemas en esta funcion

  Ultima revision: 10-09-06

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ProbInvGeoElip.m**:

```
  Funcion:    geo_ProbInvGeoElip.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Resuelve el problema inverso de la geodesia sobre la esfera por el
              metodo de Bessel modificado y calcula una estimacion del error
              cometido (varianza)
              La formulacion ha sido extraida de [GA-HL-GEO], pags. 215-232

  Entradas:   - Latitud geodesica del punto de partida (escalar o vector
                columna), en radianes
              - Longitud geodesica del punto de partida (escalar o vector
                columna), en radianes
              - Latitud geodesica del punto de llegada (escalar o vector
                columna), en radianes
              - Longitud geodesica del punto de llegada (escalar o vector
```

```
                columna), en radianes
              - Semieje mayor del elipsoide
              - Aplanamiento del elipsoide, (a-b)/a
              Argumentos opcionales:
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza del punto de partida. Dos posibilidades:
                  - Vector o matriz de dos columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto. Queda asi
                    expresada (por filas) una matriz varianza-covarianza
                    diagonal:
                    varCov=[Slat^2 Slon^2];
                  - Vector o matriz de tres columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Slat^2 Slatlon Slon^2];
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza del punto de llegada. Las mismas
                  posibilidades que en el caso de las matrices de los puntos de
                  partida
                Las varianzas han de ser pasadas en valor absoluto (no se
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envian tres
                variables de valor -1, no se hace el calculo del error

Salidas:      - Longitud de la linea geodesica (escalar o vector columna), en
                las mismas unidades que el semieje mayor del elipsoide
              - Acimut directo (del punto de partida al de llegada) de la linea
                geodesica (escalar o vector columna), en radianes
              - Acimut reciproco (del punto de llegada al de partida) de la
                linea geodesica (escalar o vector columna), en radianes
              - Varianza de la longitud de la linea geodesica (escalar o vector
                columna)
              - Varianza del acimut directo de la linea geodesica (escalar o
                vector columna), en radianes^2
              - Varianza del acimut reciproco de la linea geodesica (escalar o
                vector columna), en radianes^2
                En el caso de no haber pasado datos para el calculo del error,
                los argumentos de salida referentes a las varianzas tienen el
                valor -1 (si se han solicitado)

Ejemplo:      [dist12,aci12,aci21]=geo_ProbInvGeoElip(lat1,lon1,lat2,lon2,a,f);
              [dist12,aci12,aci21]=geo_ProbInvGeoElip(lat1,lon1,lat2,lon2,a,f,
              -1,-1,-1);
              [dist12,aci12,aci21,varDist,varAci12,varAci21]=geo_ProbInvGeoElip(
              lat1,lon1,lat2,lon2,a,f,varCovPto1,varCovPto2);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoLonGeo.m
                             - geo_ControlDominioAF.m
                             - geo_DesactivaControles.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_RestauraControles.m
                             - geo_VectorMatrizSimInterna.m
                             - geo_ProbInvGeoEsf.m
                             - geo_ProbDirGeoEsf.m
```

```
                              - geo_PonAcimutDominio.m

Funciones relacionadas de GEO: - geo_ProbDirGeo.m
                               - geo_ProbDirGeoElip.m
                               - geo_ProbDirGeoEsf.m
                               - geo_ProbInvGeo.m
                               - geo_ProbInvGeoElip.m

Nota: Esta funcion da resultados inconsistentes en el calculo del error de los
      acimutes directo y reciproco. Puede ser debido a inestabilidades en el
      calculo de las derivadas numericas

Ultima revision: 20-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_RadioElipMer.m**:

```
Funcion:    geo_RadioElipMer.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula el radio de curvatura de la elipse meridiana a una latitud
            dada y una estimacion de su error (varianza)

Entradas:   - Latitud geodesica (escalar o vector columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              La varianza ha de ser pasada en valor absoluto (no se controla
              internamente)
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Radio de curvatura de la elipse meridiana (escalar o vector
              columna), en las mismas unidades que las del semieje mayor
              pasado
            - Estimacion de la varianza, Sro^2, del radio calculado, en las
              mismas unidades (al cuadrado) que las del semieje mayor pasado
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [ro]=geo_RadioElipMer(latitudGeodesica,a,f);
            [ro]=geo_RadioElipMer(latitudGeodesica,a,f,-1);
            [ro,varRo]=geo_RadioElipMer(latitudGeodesica,a,f,varLatGeo);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoAF.m

Funciones relacionadas de GEO: - geo_RadioParalelo.m
                               - geo_RadioPrimVert.m

Ultima revision: 19-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_RadioParalelo.m**:

```
Funcion:    geo_RadioParalelo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula el radio de curvatura del paralelo a una latitud dada  y
            una estimacion de su error (varianza)

Entradas:   - Latitud geodesica (escalar o vector columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              La varianza ha de ser pasada en valor absoluto (no se controla
              internamente)
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Radio de curvatura del paralelo (escalar o vector columna), en
              las mismas unidades que las del semieje mayor pasado
            - Estimacion de la varianza, Srp^2, del radio calculado, en las
              mismas unidades (al cuadrado) que las del semieje mayor pasado
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [radPar]=geo_RadioParaleloError(latGeo,a,f);
            [radPar]=geo_RadioParaleloError(latGeo,a,f,-1);
            [radPar,varRadPar]=geo_RadioParaleloError(latGeo,a,f,varLatGeo);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoAF.m

Funciones relacionadas de GEO: - geo_RadioElipMer.m
                               - geo_RadioPrimVert.m

Ultima revision: 19-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_RadioPrimVert.m**:

```
Funcion:    geo_RadioPrimVert.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula el radio de curvatura del primer vertical a una latitud
            dada y una estimacion de su error (varianza)

Entradas:   - Latitud geodesica (escalar o vector columna), en radianes
            - Semieje mayor del elipsoide
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
```

```
              La varianza ha de ser pasada en valor absoluto (no se controla
              internamente)
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Radio de curvatura del primer vertical (escalar o vector
              columna), en las mismas unidades que las del semieje mayor
              pasado
            - Estimacion de la varianza, Snu^2, del radio calculado, en las
              mismas unidades (al cuadrado) que las del semieje mayor pasado
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [nu]=geo_RadioPrimVert(latitudGeodesica,a,f);
            [nu]=geo_RadioPrimVert(latitudGeodesica,a,f,-1);
            [nu,varNu]=geo_RadioPrimVert(latitudGeodesica,a,f,varLatGeo);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoAF.m

Funciones relacionadas de GEO: - geo_RadioElipMer.m
                               - geo_RadioParalelo.m

Ultima revision: 19-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

# Capítulo 15
# Sección 07-proyCarto

En esta carpeta se almacenan las funciones relacionadas con el trabajo con proyecciones cartográficas.

## 15.1. Carpeta 01-estructuraProyCarto

En esta carpeta se almacenan las funciones relacionadas con el trabajo con la creación de la estructura proyección cartográfica.

## 15.2. Carpeta 02-transformacionProyCarto

En esta carpeta se almacenan las funciones relacionadas con la transformación de coordenadas a cualquier proyección cartográfica.

## 15.3. Carpeta 03-proyecciones

En esta carpeta se almacenan las funciones definidoras de las proyecciones cartográficas.

### 15.3.1. Carpeta estereografica

- **geo_EstereograficaGeo.m**:

```
Funcion:    geo_EstereograficaGeo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Pasa de coordenadas cartesianas en proyeccion estereografica a
            coordenadas geodesicas y calcula una estimacion del error cometido
            (matriz varianza-covarianza)
            La formulacion ha sido extraida de [SNY-WM-2003], pags. 158-159
            161-162

Entradas:   - Coordenada X del punto de trabajo (escalar o vector columna)
            - Coordenada Y del punto de trabajo (escalar o vector columna)
            - Latitud geodesica origen (punto de tangencia del plano), en
              radianes
            - Longitud geodesica origen (punto de tangencia del plano), en
              radianes
            - Factor de escala en el origen (punto de tangencia del plano), en
              unidades adimensionales
```

```
            - Traslacion en la coordenada X, en las mismas unidades que las
              coordenadas
            - Traslacion en la coordenada Y, en las mismas unidades que las
              coordenadas
            - Semieje mayor del elipsoide, en las mismas unidades que las
              coordenadas
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto de trabajo. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sx^2 Sy^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sx^2 Sxy Sy^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si trabajamos con matrices varianza-covarianza diagonales, la
              salida sera tambien una matriz diagonal, considerando las
              coordenadas independientes. Para hacer una propagacion de
              errores estricta se utilizara como argumento de entrada una
              matriz varianza-covarianza completa (con ceros fuera de la
              diagonal principal si la matriz de entrada es diagonal)
              Si, en vez de las matrices varianza-covarianza, se envia una
              variable de valor -1, no se hace el calculo del error

Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Matriz varianza-covarianza del punto de trabajo, en radianes al
              cuadrado
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Slat^2 Slon^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Slat^2 Slatlon Slon^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [lat,lon]=geo_EstereograficaGeo(X,Y,lat0,lon0,sigma,Tx,Ty,a,f);
            [lat,lon]=geo_EstereograficaGeo(X,Y,lat0,lon0,sigma,Tx,Ty,a,f,-1);
            [lat,lon,varCov]=geo_EstereograficaGeo(X,Y,lat0,lon0,sigma,Tx,Ty,
            a,f,varCov);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominio.m
                             - geo_ControlDominioAF.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_LatConfLatGeo.m
```

```
                                - geo_LatGeoLatConf.m
                                - geo_RestauraControles.m
                                - geo_PonLonGeoDominio.m

    Funciones relacionadas de GEO: - geo_GeoEstereografica.m

    Nota: Calcula las derivadas necesarias de forma numerica. Pueden presentarse
          problemas numericos en el calculo de la propagacion de errores con
          puntos cuya longitud este en el entorno de pi

    Ultima revision: 09-02-08

    Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GeoEstereografica.m**:

```
    Funcion:    geo_GeoEstereografica.m

    Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

    Proposito:  Pasa de coordenadas geodesicas a coordenadas cartesianas en
                proyeccion estereografica y calcula una estimacion del error
                cometido (matriz varianza-covarianza)
                La formulacion ha sido extraida de [SNY-WM-2003], pags. 157-158
                160-161

    Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
                  columna), en radianes
                - Longitud geodesica del punto de trabajo (escalar o vector
                  columna), en radianes
                - Latitud geodesica origen (punto de tangencia del plano), en
                  radianes
                - Longitud geodesica origen (punto de tangencia del plano), en
                  radianes
                - Factor de escala en el origen (punto de tangencia del plano), en
                  unidades adimensionales
                - Traslacion en la coordenada X
                - Traslacion en la coordenada Y
                - Semieje mayor del elipsoide, en las mismas unidades que las
                  traslaciones
                - Aplanamiento del elipsoide, (a-b)/a
                Argumento opcional:
                  - Vector o matriz que contiene informacion sobre la matriz
                    varianza-covarianza del punto de trabajo. Dos posibilidades:
                    - Vector (o matriz) de dos columnas, cada una de las cuales
                      contiene la varianza de cada coordenada del punto. Queda asi
                      expresada una matriz varianza-covarianza diagonal:
                      varCov=[Slat^2 Slon^2];
                    - Vector (o matriz) de tres columnas, cada una de las cuales
                      contiene la varianza de cada coordenada del punto y sus
                      correlaciones. Queda asi expresada (por filas) una matriz
                      varianza-covarianza completa:
                      varCov=[Slat^2 Slatlon Slon^2];
                  Las varianzas han de ser pasadas en valor absoluto (no se
                  controla internamente)
                  Si trabajamos con matrices varianza-covarianza diagonales, la
                  salida sera tambien una matriz diagonal, considerando las
                  coordenadas independientes. Para hacer una propagacion de
```

```
                errores estricta se utilizara como argumento de entrada una
                matriz varianza-covarianza completa (con ceros fuera de la
                diagonal principal si la matriz de entrada es diagonal)
                Si, en vez de las matrices varianza-covarianza, se envia una
                variable de valor -1, no se hace el calculo del error

Salidas:    - Coordenada X proyectada (escalar o vector columna), en las
                mismas unidades que las de las traslaciones
            - Coordenada Y proyectada (escalar o vector columna), en las
                mismas unidades que las de las traslaciones
            - Matriz varianza-covarianza del punto proyectado, en las mismas
                unidades (al cuadrado para las varianzas) que las de las
                traslaciones
                Si en la entrada se ha utilizado una matriz diagonal, la salida
                tambien sera diagonal, de la forma:
                varCov=[Sx^2 Sy^2];
                Si en la entrada se ha utilizado una matriz completa, la salida
                tambien sera completa, de la forma:
                varCov=[Sx^2 Sxy Sy^2];
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [X,Y]=geo_GeoEstereografica(lat,lon,lat0,lon0,sigma,Tx,Ty,a,f);
            [X,Y]=geo_GeoEstereografica(lat,lon,lat0,lon0,sigma,Tx,Ty,a,f,-1);
            [X,Y,varCov]=geo_GeoEstereografica(lat,lon,lat0,lon0,sigma,Tx,Ty,
            a,f,varCov);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoLonGeo.m
                             - geo_ControlDominio.m
                             - geo_ControlDominioAF.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_LatGeoLatConf.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_EstereograficaGeo.m

Nota: Calcula las derivadas necesarias de forma numerica

Ultima revision: 16-11-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

### 15.3.2. Carpeta tranMerc

- **geo_GeoTranMerc.m**:

```
Funcion:    geo_GeoTranMerc.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Pasa de coordenadas geodesicas a coordenadas cartesianas en
            proyeccion transversa de Mercator (Gauss-Kruger) y calcula una
            estimacion del error cometido (matriz varianza-covarianza)
```

```
Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Latitud geodesica del paralelo origen de la proyeccion, en
              radianes
            - Longitud geodesica del meridiano central de la proyeccion, en
              radianes
            - Factor de escala a lo largo del meridiano central de la
              proyeccion, en unidades adimensionales
            - Traslacion en la coordenada X
            - Traslacion en la coordenada Y
            - Semieje mayor del elipsoide, en las mismas unidades que las
              traslaciones
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto de trabajo. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Slat^2 Slon^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Slat^2 Slatlon Slon^2];
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si trabajamos con matrices varianza-covarianza diagonales, la
              salida sera tambien una matriz diagonal, considerando las
              coordenadas independientes. Para hacer una propagacion de
              errores estricta se utilizara como argumento de entrada una
              matriz varianza-covarianza completa (con ceros fuera de la
              diagonal principal si la matriz de entrada es diagonal)
              Si, en vez de las matrices varianza-covarianza, se envia una
              variable de valor -1, no se hace el calculo del error

Salidas:    - Coordenada X proyectada (escalar o vector columna), en las
              mismas unidades que las de las traslaciones
            - Coordenada Y proyectada (escalar o vector columna), en las
              mismas unidades que las de las traslaciones
            - Matriz varianza-covarianza del punto proyectado, en las mismas
              unidades (al cuadrado para las varianzas) que las de las
              traslaciones
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Sx^2 Sy^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Sx^2 Sxy Sy^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [X,Y]=geo_GeoTranMerc(lat,lon,lat0,lon0,sigma,Tx,Ty,a,f);
            [X,Y]=geo_GeoTranMerc(lat,lon,lat0,lon0,sigma,Tx,Ty,a,f,-1);
            [X,Y,varCov]=geo_GeoTranMerc(lat,lon,lat0,lon0,sigma,Tx,Ty,a,f,
```

```
            varCov);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoLonGeo.m
                             - geo_ControlDominio.m
                             - geo_ControlDominioAF.m
                             - geo_Find.m
                             - geo_DesactivaControles.m
                             - geo_RadioPrimVert.m
                             - geo_LatGeoDistMer.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_TranMercGeo.m

Nota: Calcula las derivadas necesarias de forma numerica. La formulacion
      empleada garantiza en latitudes medias una precision de unos 15mm en la
      proyeccion de un punto con un incremento de longitud de 15 grados
      sexagesimales con respecto al meridiano origen

Ultima revision: 10-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_TranMercGeo.m**:

```
Funcion:    geo_TranMercGeo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Pasa de coordenadas cartesianas en proyeccion transversa de
            Mercator (Gauss-Kruger) a coordenadas geodesicas y calcula una
            estimacion del error cometido (matriz varianza-covarianza)

Entradas:   - Coordenada X del punto de trabajo (escalar o vector columna)
            - Coordenada Y del punto de trabajo (escalar o vector columna)
            - Latitud geodesica del paralelo origen de la proyeccion, en
              radianes
            - Longitud geodesica del meridiano central de la proyeccion, en
              radianes
            - Factor de escala a lo largo del meridiano central de la
              proyeccion, en unidades adimensionales
            - Traslacion en la coordenada X
            - Traslacion en la coordenada Y
            - Semieje mayor del elipsoide, en las mismas unidades que las
              traslaciones
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Vector o matriz que contiene informacion sobre la matriz
                varianza-covarianza del punto de trabajo. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sx^2 Sy^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
```

```
                varCov=[Sx^2 Sxy Sy^2];
            Las varianzas han de ser pasadas en valor absoluto (no se
            controla internamente)
            Si trabajamos con matrices varianza-covarianza diagonales, la
            salida sera tambien una matriz diagonal, considerando las
            coordenadas independientes. Para hacer una propagacion de
            errores estricta se utilizara como argumento de entrada una
            matriz varianza-covarianza completa (con ceros fuera de la
            diagonal principal si la matriz de entrada es diagonal)
            Si, en vez de las matrices varianza-covarianza, se envia una
            variable de valor -1, no se hace el calculo del error

Salidas:    - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Longitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Matriz varianza-covarianza del punto de trabajo, en radianes al
              cuadrado
              Si en la entrada se ha utilizado una matriz diagonal, la salida
              tambien sera diagonal, de la forma:
              varCov=[Slat^2 Slon^2];
              Si en la entrada se ha utilizado una matriz completa, la salida
              tambien sera completa, de la forma:
              varCov=[Slat^2 Slatlon Slon^2];
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [lat,lon]=geo_TranMercGeo(X,Y,lat0,lon0,sigma,Tx,Ty,a,f);
            [lat,lon]=geo_TranMercGeo(X,Y,lat0,lon0,sigma,Tx,Ty,a,f,-1);
            [lat,lon,varCov]=geo_TranMercGeo(X,Y,lat0,lon0,sigma,Tx,Ty,a,f,
            varCov);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoLonGeo.m
                             - geo_ControlDominio.m
                             - geo_ControlDominioAF.m
                             - geo_DesactivaControles.m
                             - geo_LatGeoDistMer.m
                             - geo_DistMerLatGeo.m
                             - geo_RadioPrimVert.m
                             - geo_RestauraControles.m
                             - geo_PonLonGeoDominio.m

Funciones relacionadas de GEO: - geo_GeoTranMerc.m

Nota: Calcula las derivadas necesarias de forma numerica. La formulacion
      empleada garantiza en latitudes medias una precision de unos 15mm en la
      proyeccion de un punto con un incremento de longitud de 15 grados
      sexagesimales con respecto al meridiano origen

Ultima revision: 10-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

# Capítulo 16
# Sección 08-redesGeodesicas

En esta carpeta se almacenan las funciones relacionadas con el trabajo con redes geodésicas.

## 16.1. Carpeta 01-disenyoRedes

En esta carpeta se almacenan funciones relacionadas con los distintos problemas de diseño de redes geodésicas.

### 16.1.1. Carpeta 01-generacionObsRedes

En esta carpeta se almacenan las funciones relacionadas con la generación de observaciones en redes geodésicas.

- **geo_GenRedGPS.m**:

```
Funcion:    geo_GenRedGPS.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Genera observaciones para una red GPS

Entradas:   - Matriz de puntos de la red, de nx4, con la siguiente estructura:
              obs=[N1  X1  Y1  Z1
                   N2  X2  Y2  Z2
                   N3  X3  Y3  Z3];
                 Cada fila corresponde a un punto:
                 - Col. 1: Numero identificador de cada punto
                 - Col. 2: Coordenada X geocentrica de cada punto
                 - Col. 3: Coordenada Y geocentrica de cada punto
                 - Col. 4: Coordenada Z geocentrica de cada punto
              Las coordenadas de los puntos dato han de estar en la misma
              magnitud lineal
            - Nombre del fichero de salida de resultados, como cadena de texto
            Argumentos opcionales:
              - Estructura instrumento

Salidas:    - Fichero de observaciones con todas las combinaciones posibles

Ejemplo:    estado=geo_GenRedGPS(puntos,nombreFichero);
            estado=geo_GenRedGPS(puntos,nombreFichero,A,Bppm);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_EsEscalar.m
```

```
                                - geo_EsInstrumento.m

    Funciones relacionadas de GEO: - geo_GenRedPlanoCla.m

    Ultima revision: 11-07-06

    Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GenRedPlanoCla.m**:

## 16.2. Carpeta 02-analisisRedes

En esta carpeta se guardan funciones relacionadas con el analisis de las relaciones entre los puntos fijos, libres y las observaciones que conforman una red geodesica.

- **geo_AnalisisRedGps.m**:

```
    ************************************************************************
    ESTA FUNCION SOLO ESTA PROYECTADA. NO ES FUNCIONAL
    ************************************************************************

    Funcion:    geo_AnalisisRedGps.m

    Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

    Proposito:  Realiza un analisis de los puntos fijos utilizados, puntos libres
                calculables y observaciones utilizadas en una red GPS a partir de
                los datos de partida originales

    Entradas:   - Matriz de observaciones, de tamano Nx5, con la siguiente
                  estructura:
                  obs=[ EST1  VIS1  dX1  dY1  dZ1
                       ......................
                        ESTn  VISn  dXn  dYn  dZn];
                  Cada fila corresponde a una observacion:
                  - Col. 1: Numero identificador del punto estacion
                  - Col. 2: Numero identificador del punto visado
                  - Col. 3: Componente X del vector GPS
                  - Col. 4: Componente Y del vector GPS
                  - Col. 5: Componente Z del vector GPS
                  Los valores de todas las observaciones han de estar en la misma
                  magnitud lineal
                - Matriz de coordenadas de los puntos fijos de la red, de tamano
                  Mx4, con la siguiente estructura:
                  coorFijos=[ NF1  X1  Y1  Z1
                             ..............
                              NFm  Xm  Ym  Zm];
                  Cada fila corresponde a un punto:
                  - Col. 1: Numero identificador del punto
                  - Col. 2: Coordenada X cartesiana geocentrica del punto
                  - Col. 3: Coordenada Y cartesiana geocentrica del punto
                  - Col. 4: Coordenada Z cartesiana geocentrica del punto
                  Los valores de todas las coordenadas han de estar en la misma
```

```
                magnitud lineal
              - Nombre del proyecto de red GPS, como cadena de texto
              Argumentos opcionales:
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza de las observaciones. Dos posibilidades:
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada componente del vector. Queda
                    asi expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sdx^2 Sdy^2 Sdz^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada componente del vector y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sdx^2 Sdxdy Sdxdz Sdy^2 Sdydz Sdz^2];
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza de las coordenadas de los puntos fijos
                  Dos posibilidades:
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada componente de posicion. Queda
                    asi expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sx^2 Sy^2 Sz^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada componente de posicion y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
                Ha de coincidir el numero de columnas de las matrices de
                varianza-covarianza de las observaciones y los puntos fijos
                Las varianzas han de ser pasadas en valor absoluto (no se
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envian dos
                variables de valor -1, se considera que ni las observaciones ni
                las coordenadas de los puntos fijos tienen incertidumbre

Salidas:      - Matriz de visuales, de tamano (Ptos.Libres)x(2+3*numObs), con la
                siguiente estructura:
                visuales = [N1  NVis  PosFijos  PosObs  Dir/inv  ...
                            ....................................
                            Nn  NVis  PosFijos  PosObs  Dir/Inv  ...];
                Cada fila corresponde a un punto:
                - Col. 1: Numero identificador del punto libre
                - Col. 2: Numero de visuales posibles para su calculo
                  A partir de aqui, por cada visual posible vienen grupos de
                  tres elementos que definen cada una de ellas
                  - Elem. 1: Posicion en la matriz de estado de puntos del punto
                             fijo de la visual
                  - Elem. 2: Posicion en la matriz de observaciones de la
                             observacion en curso
                  - Elem. 3: Numero indicador del sentido de la visual:
                             directa/inversa -> 1/-1
                             Que la visual sea directa significa que el punto
                             libre actual es el punto visado en la visual y que
                             sea inversa significa que el punto libre es el
                             punto estacion
                Las zonas de la matriz que no contienen datos de visuales
                almacenan el valor 0, excepto en el campo numero identificador
                de punto libre, donde siempre existe dato
```

```
              El orden de las filas de la matriz es el mismo que el de los
              puntos libres en el vector de entrada
            - Si la funcion no encuentra ninguna visual valida devuelve el
              valor -1

Ejemplo:    [visuales]=geo_AnalisisRedGps(fijos,libres,obs);

Funciones utilizadas de GEO: - geo_EsTexto.m
                             - geo_BuscaNumero.m

Funciones relacionadas de GEO: - geo_CalculoRedClasicaElipsoide.m

Ultima revision: 05-01-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 16.3. Carpeta 03-calculoRedes

En esta carpeta se almacenan las funciones relacionadas con el cálculo de redes geodésicas.

- **geo_CalculoRedGps.m**:

```
Funcion:    geo_CalculoRedGps.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calculo de coordenadas aproximadas de una red GPS

Entradas:   - Matriz de observaciones, de tamano Nx5, con la siguiente
              estructura:
              obs=[ EST1  VIS1  dX1  dY1  dZ1
                   ......................
                    ESTn  VISn  dXn  dYn  dZn];
              Cada fila corresponde a una observacion:
              - Col. 1: Numero identificador del punto estacion
              - Col. 2: Numero identificador del punto visado
              - Col. 3: Componente X del vector GPS
              - Col. 4: Componente Y del vector GPS
              - Col. 5: Componente Z del vector GPS
              Los valores de todas las observaciones han de estar en la misma
              magnitud lineal
            - Matriz de coordenadas de los puntos fijos de la red, de tamano
              Mx4, con la siguiente estructura:
              coorFijos=[ NF1  X1  Y1  Z1
                          ..............
                           NFn  Xm  Ym  Zm];
              Cada fila corresponde a un punto:
              - Col. 1: Numero identificador del punto
              - Col. 2: Coordenada X cartesiana geocentrica del punto
              - Col. 3: Coordenada Y cartesiana geocentrica del punto
              - Col. 4: Coordenada Z cartesiana geocentrica del punto
              Los valores de todas las coordenadas han de estar en la misma
              magnitud lineal
            - Nombre del fichero de salida de resultados, como cadena de texto
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de las observaciones. Dos posibilidades:
```

```
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada componente del vector. Queda
                  asi expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sdx^2 Sdy^2 Sdz^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada componente del vector y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sdx^2 Sdxdy Sdxdz Sdy^2 Sdydz Sdz^2];
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de las coordenadas de los puntos fijos
                Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada componente de posicion. Queda
                  asi expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sx^2 Sy^2 Sz^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada componente de posicion y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
              Ha de coincidir el numero de columnas de las matrices de
              varianza-covarianza de las observaciones y los puntos fijos
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, se considera que ni las observaciones ni
              las coordenadas de los puntos fijos tienen incertidumbre

Salidas:      - Matriz de coordenadas de los puntos incognita de la red, de
                tamano nx4, con la siguiente estructura:
                coorFijos=[ NI1  X1  Y1  Z1
                            ..............
                             NIn  Xn  Yn  Zn];
                Cada fila corresponde a un punto:
                - Col. 1: Numero identificador del punto
                - Col. 2: Coordenada X cartesiana geocentrica del punto
                - Col. 3: Coordenada Y cartesiana geocentrica del punto
                - Col. 4: Coordenada Z cartesiana geocentrica del punto
              - Matrices varianza-covarianza de los puntos incognita, en las
                mismas unidades (al cuadrado para las varianzas) que las de las
                coordenadas de los puntos fijos. Cada fila de la matriz es una
                matriz varianza-covarianza del punto incognita correspondiente
                a la misma posicion de la matriz de coordenadas de los puntos
                Si en la entrada se ha utilizado una matriz diagonal, la salida
                tambien sera diagonal, de la forma:
                varCov=[Sx^2 Sy^2 Sz^2];
                Si en la entrada se ha utilizado una matriz completa, la salida
                tambien sera completa, de la forma:
                varCov=[Sdx^2 Sdxdy Sdxdz Sdy^2 Sdydz Sdz^2];
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)
              - Fichero de resultados, con un resumen de los calculos hechos y
                las coordenadas finales. Este fichero se guarda en el directorio
                de trabajo

Ejemplo:      [coorPtos]=geo_CalculoRedGps(obs,fijos,nomFich);
```

```
            [coorPtos,varianza]=geo_CalculoRedGps(obs,fijos,nomFich,-1,-1);
            [coorPtos,varianza]=geo_CalculoRedGps(obs,fijos,nomFich,varCovObs,
            varCovFijos);

  Funciones utilizadas de GEO: - geo_EsTexto.m
                               - geo_HayQueCalcularErrores.m
                               - geo_DesactivaControles.m
                               - geo_AjustaFilasVectorMatriz.m
                               - geo_PuntosLibresRed.m
                               - geo_Redimensiona.m
                               - geo_Fecha.m
                               - geo_NumeroVisualesRedGps.m
                               - geo_OrdenaMatriz.m
                               - geo_MediaCoordenadas.m
                               - geo_RestauraControles.m

  Funciones relacionadas de GEO: - geo_CalculoRedClasicaElipsoide.m

  Ultima revision: 09-02-08

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_NumeroVisualesRedGps.m**:

```
  Funcion:    geo_NumeroVisualesRedGps.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Calcula el numero de visuales posibles para el calculo de un punto
              incognita en una red GPS

  Entradas:   - Vector columna que contiene los numeros identificativos de los
                puntos fijos que intervienen en la red. Cada fila es un punto
              - Vector columna que contiene los numeros identificativos de los
                puntos libres que intervienen en la red. Cada fila es un punto
              - Matriz de observaciones, de tamano Nx2, con la siguiente
                estructura:
                obs=[ EST1  VIS1
                      ...........
                       ESTn  VISn];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador del punto estacion
                - Col. 2: Numero identificador del punto visado

  Salidas:    - Matriz de visuales, de tamano (Ptos.Libres)x(2+3*numObs), con la
                siguiente estructura:
                visuales = [N1  NVis  PosFijos  PosObs  Dir/inv  ...
                            ....................................
                            Nn  NVis  PosFijos  PosObs  Dir/Inv  ...];
                Cada fila corresponde a un punto:
                - Col. 1: Numero identificador del punto libre
                - Col. 2: Numero de visuales posibles para su calculo
                  A partir de aqui, por cada visual posible vienen grupos de
                  tres elementos que definen cada una de ellas
                  - Elem. 1: Posicion en la matriz de estado de puntos del punto
                             fijo de la visual
                  - Elem. 2: Posicion en la matriz de observaciones de la
                             observacion en curso
```

```
                - Elem. 3: Numero indicador del sentido de la visual:
                           directa/inversa -> 1/-1
                           Que la visual sea directa significa que el punto
                           libre actual es el punto visado en la visual y que
                           sea inversa significa que el punto libre es el
                           punto estacion
              Las zonas de la matriz que no contienen datos de visuales
              almacenan el valor 0, excepto en el campo numero identificador
              de punto libre, donde siempre existe dato
              El orden de las filas de la matriz es el mismo que el de los
              puntos libres en el vector de entrada
            - Si la funcion no encuentra ninguna visual valida devuelve el
              valor -1

Ejemplo:    [visuales]=geo_NumeroVisualesRedGps(fijos,libres,obs);

Funciones utilizadas de GEO: - geo_DesactivaControles.m
                             - geo_BuscaNumero.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_CalculoRedClasicaElipsoide.m

Ultima revision: 26-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PuntosLibresRed.m**:

```
Funcion:    geo_PuntosLibresRed.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Busca los puntos libres en una red geodesica

Entradas:   - Vector columna de puntos fijos de la red, con la siguiente
              estructura:
              fijos=[ NF1
                      ....
                      NFn];
              Cada fila corresponde a un punto:
              - Col. 1: Numero identificador del punto
              En el desarrollo de la funcion se consideran fijos todos los
              puntos de la matriz pasada. Esto es asi para permitir la
              utilizacion de la funcion en modo secuencial: si, por ejemplo,
              tenemos una red en la que hay observaciones de distancia y
              angulares, primero determinaremos los puntos libres comparando,
              por ejemplo, la matriz de puntos fijos original con las
              observaciones de distancia pero, a la hora de pasar las
              observaciones angulares los puntos libres detectados en el paso
              anterior se consideraran ahora fijos para evitar repeticiones
            - Matriz de observaciones, de tamano NxK, con la siguiente
              estructura:
              obs=[ N11  N12  ...  N1k
                    ................
                    Nn1  Nn2  ...  Nnk];
              Cada fila corresponde a una observacion:
              - Col. 1: Numero identificador del punto que interviene en la
                        observacion
```

```
                - Col. 2: Numero identificador del punto que interviene en la
                          observacion
                  .......
                - Col. k: Numero identificador del punto que interviene en la
                          observacion
                El hecho de que la matriz de observaciones tenga un numero no
                definido de columnas implica que esta funcion puede trabajar con
                varios tipos de observaciones como, por ejemplo: observaciones
                GPS (2 puntos implicados: estacion y visado), observaciones
                angulares (3 puntos implicados), etc

Salidas:      - Vector columna de puntos fijos original, ampliado con los puntos
                libres extraidos de las observaciones, con la siguiente
                estructura:
                nuevosFijos=[ NF1
                              ....
                               NFn
                               NL1
                              ....
                               NLm];
                Cada fila corresponde a un punto:
                - Col. 1: Numero identificador del punto

Ejemplo:      nuevosFijos=geo_PuntosLibresRed(fijos,observaciones);

Funciones utilizadas de GEO: - geo_DesactivaControles.m
                             - geo_BuscaNumero.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_ObservacionesLibresRed.m

Ultima revision: 26-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 16.4. Carpeta 04-compensacionRedes

En esta carpeta se almacenan las funciones relacionadas con la compensación de redes geodésicas.

- **geo_CreaMatricesCompRedGps.m**:

```
************************************************************
ATENCION: ESTA FUNCION NO ESTA TERMINADA
************************************************************
************************************************************
ATENCION: ESTA FUNCION NO ESTA TERMINADA
************************************************************

Funcion:      geo_CreaMatricesCompRedGps.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:    Crea las matrices necesarias para la compensacion MMCC de una red
              GPS

Entradas:     - Matriz de observaciones, de tamano Nx5, con la siguiente
```

```
                estructura:
                obs=[ EST1  VIS1  dX1  dY1  dZ1
                     ......................
                      ESTn  VISn  dXn  dYn  dZn];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador del punto estacion
                - Col. 2: Numero identificador del punto visado
                - Col. 3: Componente X del vector GPS
                - Col. 4: Componente Y del vector GPS
                - Col. 5: Componente Z del vector GPS
                Los valores de todas las observaciones han de estar en la misma
                magnitud lineal
              - Matriz de coordenadas de los puntos fijos de la red, de tamano
                nx4, con la siguiente estructura:
                coorFijos=[ NF1  X1  Y1  Z1
                           ..............
                            NFn  Xn  Yn  Zn];
                Cada fila corresponde a un punto:
                - Col. 1: Numero identificador del punto
                - Col. 2: Coordenada X cartesiana geocentrica del punto
                - Col. 3: Coordenada Y cartesiana geocentrica del punto
                - Col. 4: Coordenada Z cartesiana geocentrica del punto
                Los valores de todas las coordenadas han de estar en la misma
                magnitud lineal
              Argumentos opcionales:
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza de las observaciones. Dos posibilidades:
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada componente del vector. Queda
                    asi expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sdx^2 Sdy^2 Sdz^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada componente del vector y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sdx^2 Sdxdy Sdxdz Sdy^2 Sdydz Sdz^2];
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza de las coordenadas de los puntos fijos
                  Dos posibilidades:
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada componente de posicion. Queda
                    asi expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sx^2 Sy^2 Sz^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada componente de posicion y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sx^2 Sxy Sxz Sy^2 Syz Sz^2];
                Ha de coincidir el numero de columnas de las matrices de
                varianza-covarianza de las observaciones y los puntos fijos
                Las varianzas han de ser pasadas en valor absoluto (no se
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envian dos
                variables de valor -1, se considera que ni las observaciones ni
                las coordenadas de los puntos fijos tienen incertidumbre

Salidas:      - Matriz de diseno, A, del sistema A*x=t
```

```
            - Termino independiente, t, del sistema A*x=t
            - Vector columna que contiene los identificadores de los puntos
              libres. Este sera el orden de las incognitas del vector x del
              sistema A*x=t
            - Matriz de pesos, P, del sistema A*x=t. Si en la entrada se han
              introducido matrices de varianza-covarianza diagonales, la
              salida sera un vector fila que contiene los elementos de la
              diagonal principal de la matriz P. Si se han introducido
              matrices varianza covarianza completas la salida sera una matriz
              de pesos completa
              Si no se han pasado matrices varianza-covarianza este argumento
              tendra valor -1 (si se ha solicitado)

************************************************************
ATENCION: FALTA POR CREAR LA MATRIZ DE PESOS DE LOS PUNTOS FIJOS
************************************************************

Ejemplo:    [A,t,idLibres]=geo_CalculoRedGps(obs,fijos);
            [A,t,idLibres,P]=geo_CalculoRedGps(obs,fijos,-1,-1);
            [A,t,idLibres,P]=geo_CalculoRedGps(obs,fijos,nomFich,varCovFijos);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_PuntosLibresRed.m
                             - geo_Redimensiona.m
                             - geo_Fecha.m
                             - geo_NumeroVisualesRedGps.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 05-01-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_ObservacionesLibresRed.m**:

```
Funcion:    geo_ObservacionesLibresRed.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Busca las visuales que contengan, por lo menos, un punto libre en
            una red geodesica

Entradas:   - Vector columna de puntos fijos de la red, con la siguiente
              estructura:
              fijos=[ NF1
                      ....
                      NFn];
              Cada fila corresponde a un punto:
              - Col. 1: Numero identificador del punto
            - Vector columna de puntos libres de la red, con la siguiente
              estructura:
              libres=[ NL1
                       ....
                       NLn];
              Cada fila corresponde a un punto:
              - Col. 1: Numero identificador del punto
            - Matriz de observaciones, de tamano NxK, con la siguiente
              estructura:
```

```
                obs=[ N11  N12  ...  N1k
                     ...................
                      Nn1  Nn2  ...  Nnk];
                Cada fila corresponde a una observacion:
                - Col. 1: Numero identificador del punto que interviene en la
                          observacion
                - Col. 2: Numero identificador del punto que interviene en la
                          observacion
                  .......
                - Col. k: Numero identificador del punto que interviene en la
                          observacion
                El hecho de que la matriz de observaciones tenga un numero no
                definido de columnas implica que esta funcion puede trabajar con
                varios tipos de observaciones como, por ejemplo: observaciones
                GPS (2 puntos implicados: estacion y visado), observaciones
                angulares (3 puntos implicados), etc

Salidas:    - Matriz de tamano Mx2K+1 que contiene informacion acerca de las
                visuales en las que interviene como minimo un punto libre. Su
                estructura e las siguiente:
                visLibres=[ fO1  1/0  fF11/fL11  ...  1/0  fF1k/fL1k
                            ....................................
                            fOm  1/0  fFm1/fLm1  ...  1/0  fFmk/fLmk];
                Cada fila corresponde a una visual:
                - Col. 1: Numero de fila en la matriz de observaciones pasada
                          correspondiente a la observacion actual
                  A partir de aqui, por cada observacion posible vienen grupos
                  de dos elementos que definen el estado y posicion de cada
                  punto que la compone, comenzando por el primero (en el mismo
                  orden que en la matriz de observaciones pasadas) y terminando
                  por el ultimo:
                  - Elem. 1: 1/0 -> indica si el punto es fijo/libre
                  - Elem. 2: Posicion en el vector de puntos fijos o de puntos
                             libres pasado, depende del tipo de punto fijo/libre
                Si no hay ninguna visual que implique al menos un punto libre
                este argumento de salida toma el valor -1
              - Matriz de tamano MxK+1 que contiene informacion acerca de las
                visuales en las que no interviene ningun punto libre. Su
                estructura e las siguiente:
                visFijos=[ fO1  fF11  ...  fF1k
                           ...................
                           fOm  fFm1  ...  fFmk];
                Cada fila corresponde a una visual:
                - Col. 1: Numero de fila en la matriz de observaciones pasada
                          correspondiente a la observacion actual
                  A partir de aqui, por cada observacion posible vienen grupos
                  de un elemento que definen la posicion de cada punto que la
                  compone, comenzando por el primero (en el mismo orden que en
                  la matriz de observaciones pasadas) y terminando por el
                  ultimo:
                  - Elem. 1: Posicion en el vector de puntos fijos pasado
                Si no hay ninguna visual en la que todos los puntos sean fijos
                este argumento de salida toma el valor -1

Ejemplo:    [visLibres,visFijos]=geo_ObservacionesLibresRed(fijos,libres,
            observaciones);
```

```
Funciones utilizadas de GEO: - geo_DesactivaControles.m
                             - geo_BuscaNumero.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_PuntosLibresRed.m

Ultima revision: 26-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

# Capítulo 17
# Sección 09-datum

En esta carpeta se almacenan las funciones relacionadas con los sistemas de referencia.

# Capítulo 18
# Sección 10-gravimetria

En esta carpeta se almacenan las funciones relacionadas con los cálculos gravimétricos.

## 18.1. Carpeta 01-armonicosEsfericos

En esta carpeta se almacenan las funciones relacionadas con el cálculo de polinomios y funciones asociadas de Legendre, así como con el cálculo de armónicos esféricos.

- **geo_FuncAsocLegendre.m**:

```
Funcion:    geo_FuncAsocLegendre.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula el valor de las funciones asociadas de Legendre desde
            grado y orden 0 a grado N y orden M
            La formulacion (formulas de recurrencia) ha sido extraida de
            [TO-GEOD-2001], pag. 71 y [AS-UB91-1992], pag. 25 y desarrollada a
            partir de [YU-MMAT-2006], pag. 106

Entradas:   - Grado de la funcion a calcular, numero entero >= 0
            - Orden de la funcion a calcular, numero entero <= grado
            - Argumento de la funcion (en el caso que nos ocupa, coseno de la
              colatitud esferica del punto de trabajo)
            - Cadena de texto 'UN' o 'FN' o 'QN', segun se quieran calcular
              las funciones sin normalizar (UNNORMALIZED), fuertemente
              normalizadas (FULLY-NORMALIZAED) en el sentido de
              [HM-GEFI-1985], pag. 23 o cuasi-normalizadas (QUASI-NORMALIZED)
              en el sentido de [AS-UB91-1992], pag. 25
            - Cadena de texto 'SOLO' o 'TODO' segun se quiera calcular solo
              la funcion de grado y orden indicada en la entrada o todas las
              funciones de grado y orden <= a los pasados como argumentos
              respectivamente

Salidas:    - Si se ha pasado como quinto argumento la cadena 'SOLO':
              Escalar que contiene el valor de la funcion asociada de Legendre
              indicada en la entrada para el punto de trabajo
            - Si se ha pasado como quinto argumento la cadena 'TODO':
              Matriz de (grado+1xorden+1) que contiene el valor de las
              funciones asociadas de Legendre de grado y orden <= a los
              pasados como argumentos
              fAL = [ F00
                      F10  F11
```

```
                        F20  F21  F22
                       ......................
                        FN0  FN1  FN2 ... FNM];
                  De esta forma, el elemento (i,j) de la matriz corresponde al
                  valor de la funcion asociada de grado i-1 y orden j-1
                  Los elementos de la matriz que corresponden a funciones
                  imposibles (orden > grado) contienen el valor 0

Ejemplo:     [fAsoc]=geo_FuncAsocLegendre(grado,orden,t,'UN','SOLO');
             [fAsoc]=geo_FuncAsocLegendre(grado,orden,t,'FN','TODO');
             [fAsoc]=geo_FuncAsocLegendre(grado,orden,t,'QN','TODO');

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_EsEntero.m
                             - geo_EsTexto.m
                             - geo_WarningOct.m

Funciones relacionadas de GEO: - geo_PolLegendre.m

Ultima revision: 26-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PolLegendre.m**:

```
Funcion:     geo_PolLegendre.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:   Calcula el valor de los polinomios de Legendre desde grado 0 a
             grado N
             La formulacion (formulas de recurrencia) ha sido extraida de
             [HM-GEFI-1985], pag. 23 y desarrollada a partir de [AS-UB91-1992],
             pag. 25

Entradas:    - Grado maximo del polinomio a calcular, numero entero >= 0
             - Argumento del polinomio (en el caso que nos ocupa, coseno de la
               colatitud esferica del punto de trabajo)
             - Cadena de texto 'UN' o 'FN' o 'QN', segun se quieran calcular
               los polinomios sin normalizar (UNNORMALIZED), fuertemente
               normalizados (FULLY-NORMALIZAED) en el sentido de
               [HM-GEFI-1985], pag. 23 o cuasi-normalizados (QUASI-NORMALIZED)
               en el sentido de [AS-UB91-1992], pag. 25

Salidas:     - Vector columna que contiene los valores del polinomio para el
               argumento pasado desde el grado 0 hasta el grado maximo indicado
               P = [ P0
                     P1
                    ....
                     PN];
               Cada fila corresponde a un polinomio distinto
               De esta forma, el indice de la columna en la matriz corresponde
               al grado del polinomio-1

Ejemplo:     [Pn]=geo_PolLegendre(grado,t,'UN');
             [Pn]=geo_PolLegendre(grado,t,'FN');
             [Pn]=geo_PolLegendre(grado,t,'QN');
```

```
Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_EsEntero.m
                             - geo_EsTexto.m
                             - geo_WarningOct.m

Funciones relacionadas de GEO: - geo_FuncAsocLegendre.m

Ultima revision: 26-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 18.2. Carpeta 02-gravedad

En esta carpeta se almacenan las funciones relacionadas con el cálculo de valores de la gravedad.

- **geo_GravedadCilindroVertical.m**:

```
Funcion:    geo_GravedadCilindroVertical.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la componente vertical (eje Z) de la atraccion
            gravitatoria de un conjunto de cilindros verticales sobre un
            conjunto de puntos dados situados sobre su eje (fuera o sobre la
            superficie del cilindro)
            La formulacion ha sido extraida de [HM-GEFI-1985], pag. 128

Entradas:   - Escalar o vector columna con la altitud (en metros) del punto
              atraido sobre la cara superior del cilindro
            - Vector fila o matriz de dos columnas con altura y el radio (en
              metros) del cilindro. Cada fila de la matriz es un cilindro
              distinto:
              - Col. 1: Altura del cilindro
              - Col. 2: Radio del cilindro
            - Escalar o vector columna que almacena la densidad de cada
              cilindro (en kg/m^3)
            Argumentos opcionales:
              - Varianza de la altitud sobre el cilindro del punto atraido
                (escalar o vector columna)
              - Vector (o matriz) de dos columnas, cada una de las cuales
                contiene la varianza de la altura y del radio de cada
                cilindro:
                varCov=[Sh^2 Sr^2];
              - Varianza de la densidad (escalar o vector columna)
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Escalar o vector columna con la gravedad en cada punto debida a
              la atraccion del cilindro correspondiente
            - Estimacion de la varianza, Sg^2, de la gravedad calculada para
              cada punto
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [grav]=geo_GravedadCilindroVertical(puntos,cilindro,densidades);
```

```
            [grav]=geo_GravedadCilindroVertical(puntos,cilindro,densidades,
            -1,-1,-1);
            [grav,varGrav]=geo_GravedadCilindroVertical(puntos,cilindro,
            densidades,varCovPuntos,varCilindro,varDensidades);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_Find.m

Funciones relacionadas de GEO: - geo_GravedadEsfera.m
                               - geo_GravedadPrismaMacMillan.m
                               - geo_GravedadPrismaNagy.m
                               - geo_GravedadPrismaPuntos.m
                               - geo_GravedadPunto.m
                               - geo_GravedadRectangulo.m

Nota: Las atracciones se calculan como si el punto estuviese situado encima
      del cilindro. En el caso de puntos que esten bajo el cilindro es
      responsabilidad del usuario identificarlos y cambiar el signo del valor
      de la gravedad. Esta identificacion se puede realizar mediante el truco
      de meter un valor negativo de densidad para ese caso concreto

Ultima revision: 03-03-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GravedadEsfera.m**:

```
Funcion:    geo_GravedadEsfera.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la componente vertical (eje Z) de la atraccion
            gravitatoria de un conjunto de esferas sobre un conjunto de puntos
            dados

Entradas:   - Vector fila o matriz de tres columnas con las coordenadas (en
              metros) de los puntos atraidos en un sistema cartesiano
              dextrogiro. Cada fila de la matriz es un punto distinto:
              - Col. 1: Coordenada X del punto
              - Col. 2: Coordenada Y del punto
              - Col. 3: Coordenada Z del punto
            - Vector fila o matriz de tres columnas con las coordenadas (en
              metros) de los centros de la esferas en un sistema cartesiano
              dextrogiro. Cada fila de la matriz es un centro distinto:
              - Col. 1: Coordenada X del centro de la esfera
              - Col. 2: Coordenada Y del centro de la esfera
              - Col. 3: Coordenada Z del centro de la esfera
            - Escalar o vector columna que almacena el radio de cada esfera
              Si es un vector ha de tener el mismo numero de filas que la
              matriz de coordenadas de los centros
            - Escalar o vector columna que almacena la densidad de cada esfera
              Si es un vector ha de tener el mismo numero de filas que la
              matriz de coordenadas de los centros
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos atraidos. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
```

```
                    expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sxi^2 Syi^2 Sz1^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
                - Vector o matriz que contiene informacion sobre las matrices
                  varianza-covarianza de los centros de las esferas. Dos
                  posibilidades:
                  - Vector (o matriz) de tres columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto. Queda asi
                    expresada una matriz varianza-covarianza diagonal:
                    varCov=[Sxi^2 Syi^2 Sz1^2];
                  - Vector (o matriz) de seis columnas, cada una de las cuales
                    contiene la varianza de cada coordenada del punto y sus
                    correlaciones. Queda asi expresada (por filas) una matriz
                    varianza-covarianza completa:
                    varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
                - Varianza de los radios de los puntos atrayentes (escalar o
                  vector columna)
                - Varianza de la densidad de las esferas (escalar o vector
                  columna)
                Las varianzas han de ser pasadas en valor absoluto (no se
                controla internamente)
                Si, en vez de las matrices varianza-covarianza, se envian cuatro
                variables de valor -1, no se hace el calculo del error

Salidas:    - Escalar o vector columna con la gravedad en cada punto debida a
                la atraccion de cada esfera. Si solo se ha introducido un punto
                atraido las filas del vector columna corresponden a la atraccion
                debida a cada esfera, mientras que si se han introducido varios
                puntos y varias esferas el vector columna contiene la atraccion
                debida a cada esfera sobre cada punto:
                [E1P1,...,ENP1,E1P2,...ENP2,...,E1PM,...,ENPM]^T
                donde E->Esfera y P->Punto
              - Estimacion de la varianza, Sg^2, de la gravedad calculada para
                cada punto. Escalar o vector columna con la misma estructura que
                el primer argumento de salida
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [grav]=geo_GravedadEsfera(punto,esfera,radio,densidad);
            [grav]=geo_GravedadEsfera(punto,esfera,radio,densidad,-1,-1,-1);
            [grav,varGrav]=geo_GravedadEsfera(punto,esfera,radio,densidad,
            varCovPunto,varCovEsfera,varRadiovarDensidad);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_DesactivaControles.m
                             - geo_GravedadPunto.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: - geo_GravedadCilindroVertical.m
                               - geo_GravedadPrismaMacMillan.m
                               - geo_GravedadPrismaNagy.m
                               - geo_GravedadPrismaPuntos.m
                               - geo_GravedadPunto.m
```

```
                                - geo_GravedadRectangulo.m

   Nota: No se controla internamente si los puntos atraidos estan situados en el
         interior de alguna esfera

   Ultima revision: 14-02-08

   Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GravedadPrismaMacMillan.m**:

```
   Funcion:    geo_GravedadPrismaMacMillan.m

   Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

   Proposito:  Calcula la componente vertical (eje Z) de la atraccion
               gravitatoria de un conjunto de prismas sobre un conjunto de puntos
               dados en el exterior o sobre su superficie mediante la
               aproximacion de W.D. MacMillan del potencian de un prisma
               rectangular
               La formulacion ha sido desarrollada a partir de la formula (12) de
               [HE-CTPPM-2006]

   Entradas:   - Vector fila o matriz de tres columnas con las coordenadas (en
                 metros) de los puntos atraidos en un sistema cartesiano
                 dextrogiro. Cada fila de la matriz es un punto distinto:
                 - Col. 1: Coordenada X del punto
                 - Col. 2: Coordenada Y del punto
                 - Col. 3: Coordenada Z del punto
               - Vector fila o matriz de seis columnas con las coordenadas (en
                 metros) de los limites de las caras del prisma atrayente en un
                 sistema cartesiano dextrogiro. Cada fila de la matriz es un
                 prisma distinto:
                 - Col. 1: Coordenada X1 del prisma|
                 - Col. 2: Coordenada X2 del prisma| Con (x1,y1,z1)<(x2,y2,z2)
                 - Col. 3: Coordenada Y1 del prisma| respectivamente
                 - Col. 4: Coordenada Y2 del prisma| Esta condicion no se
                 - Col. 5: Coordenada Z1 del prisma| controla internamente
                 - Col. 6: Coordenada Z2 del prisma|
               - Escalar o vector columna que almacena la densidad de cada prisma
                 (en kg/m^3). Si es un vector ha de tener el mismo numero de
                 filas que la matriz de prismas
               Argumentos opcionales:
                 - Vector o matriz que contiene informacion sobre las matrices
                   varianza-covarianza de los puntos atraidos. Dos posibilidades:
                   - Vector (o matriz) de tres columnas, cada una de las cuales
                     contiene la varianza de cada coordenada del punto. Queda asi
                     expresada una matriz varianza-covarianza diagonal:
                     varCov=[Sxi^2 Syi^2 Sz1^2];
                   - Vector (o matriz) de seis columnas, cada una de las cuales
                     contiene la varianza de cada coordenada del punto y sus
                     correlaciones. Queda asi expresada (por filas) una matriz
                     varianza-covarianza completa:
                     varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
                 - Vector (o matriz) de seis columnas, cada una de las cuales
                   contiene la varianza de cada coordenada los limites de cada
                   prisma:
                   varCov=[Sx1^2 Sx2^2 Sy1^2 Sy2^2 Sz1^2 Sz2^2];
```

```
              - Varianza de la densidad (escalar o vector columna)
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Escalar o vector columna con la gravedad en cada punto debida a
              la atraccion de cada prisma. Si solo se ha introducido un punto
              atraido las filas del vector columna corresponden a la atraccion
              debida a cada prisma, mientras que si se han introducido varios
              puntos y varios prismas el vector columna contiene la atraccion
              debida a cada prisma sobre cada punto:
              [Pr1P1,...,PrNP1,Pr1P2,...PrNP2,...,Pr1PM,...,PrNPM]^T
              donde Pr->Prisma y P->Punto
            - Estimacion de la varianza, Sg^2, de la gravedad calculada para
              cada punto. Escalar o vector columna con la misma estructura que
              el primer argumento de salida
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [grav]=geo_GravedadPrismaMacMillan(puntos,prismas,densidades);
            [grav]=geo_GravedadPrismaMacMillan(puntos,prismas,densidades,-1,
            -1,-1);
            [grav,varGrav]=geo_GravedadPrismaMacMillan(puntos,prismas,
            densidades,varCovPuntos,varPrismas,varDensidades);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_WarningOct.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_GravedadCilindroVertical.m
                               - geo_GravedadEsfera.m
                               - geo_GravedadPrismaNagy.m
                               - geo_GravedadPrismaPuntos.m
                               - geo_GravedadPunto.m
                               - geo_GravedadRectangulo.m

Nota 1: Para realizar el calculo del error en esta funcion las derivadas
        parciales correspondientes se realizan de forma numerica. El elemento
        diferencial en coordenadas para calcular las citadas derivadas se
        calcula como factor del tamano del prisma atrayente. Este factor se
        almacena en la variable global:
        geoVarGlobal.diferencial.factDimCuerpPeq
        El diferencial en densidad esta definido en
        geoVarGlobal.diferencial.densidad
        Se hace notar la importancia que tiene introducir los valores de
        densidad en las mismas unidades que las de su diferencial (kg/m^3)
        para asegurar un correcto computo del error
        Debido al hecho de calcular las derivadas en forma numerica pueden
        aparecer errores en el calculo del error porque, en casos
        particulares, el punto atraido puede quedar dentro de un prisma por la
        traslacion realizada para poder diferenciar

Nota 2: No se controla internamente si los puntos atraidos estan situados en
        el interior de algun prisma
```

```
Nota 3: La expresion para la atraccion vertical de un prisma rectangular segun
        la aproximacion de MacMillan se ha determinando derivando la expresion
        (12) de [HE-CTPPM-2006] con respecto a la coordenada Z

Ultima revision: 26-03-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GravedadPrismaNagy.m**:

```
Funcion:    geo_GravedadPrismaNagy.m

Proposito:  Calcula la componente vertical (eje Z) de la atraccion
            gravitatoria de un conjunto de prismas sobre un conjunto de puntos
            dados en el exterior o sobre su superficie
            La formulacion ha sido extraida de [NA-GRPR-2000]

Entradas:   - Vector fila o matriz de tres columnas con las coordenadas (en
              metros) de los puntos atraidos en un sistema cartesiano
              dextrogiro. Cada fila de la matriz es un punto distinto:
              - Col. 1: Coordenada X del punto
              - Col. 2: Coordenada Y del punto
              - Col. 3: Coordenada Z del punto
            - Vector fila o matriz de seis columnas con las coordenadas (en
              metros) de los limites de las caras del prisma atrayente en un
              sistema cartesiano dextrogiro. Cada fila de la matriz es un
              prisma distinto:
              - Col. 1: Coordenada X1 del prisma|
              - Col. 2: Coordenada X2 del prisma| Con (x1,y1,z1)<(x2,y2,z2)
              - Col. 3: Coordenada Y1 del prisma| respectivamente
              - Col. 4: Coordenada Y2 del prisma| Esta condicion no se
              - Col. 5: Coordenada Z1 del prisma| controla internamente
              - Col. 6: Coordenada Z2 del prisma|
            - Escalar o vector columna que almacena la densidad de cada prisma
              (en kg/m^3). Si es un vector ha de tener el mismo numero de
              filas que la matriz de prismas
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos atraidos. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2 Sz1^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
              - Vector (o matriz) de seis columnas, cada una de las cuales
                contiene la varianza de cada coordenada los limites de cada
                prisma:
                varCov=[Sx1^2 Sx2^2 Sy1^2 Sy2^2 Sz1^2 Sz2^2];
              - Varianza de la densidad (escalar o vector columna)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Escalar o vector columna con la gravedad en cada punto debida a
              la atraccion de cada prisma. Si solo se ha introducido un punto
```

```
                atraido las filas del vector columna corresponden a la atraccion
                debida a cada prisma, mientras que si se han introducido varios
                puntos y varios prismas el vector columna contiene la atraccion
                debida a cada prisma sobre cada punto:
                [Pr1P1,...,PrNP1,Pr1P2,...PrNP2,...,Pr1PM,...,PrNPM]^T
                donde Pr->Prisma y P->Punto
              - Estimacion de la varianza, Sg^2, de la gravedad calculada para
                cada punto. Escalar o vector columna con la misma estructura que
                el primer argumento de salida
                En el caso de no haber pasado datos para el calculo del error,
                este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [grav]=geo_GravedadPrismaNagy(ptos,pris,dens);
            [grav]=geo_GravedadPrismaNagy(ptos,pris,dens,-1,-1,-1);
            [grav,varGrav]=geo_GravedadPrismaNagy(ptos,pris,dens,
            varCovPtos,varPris,varDens);

Usa geoVarGlobal: Si

Nota 1:     Para realizar el calculo del error en esta funcion las derivadas
            parciales correspondientes se realizan de forma numerica. El
            elemento diferencial en coordenadas para calcular las citadas
            derivadas se calcula como factor del tamano del prisma atrayente
            Este factor se almacena en la variable global:
              geoVarGlobal.diferencial.factDimCuerpPeq
            El diferencial en densidad tambien se almacena en la variable
            global:
              geoVarGlobal.diferencial.densidad
            Se hace notar la importancia que tiene introducir los valores de
            densidad en las mismas unidades que las de su diferencial (kg/m^3)
            para asegurar un correcto computo del error
            Debido al hecho de calcular las derivadas en forma numerica pueden
            aparecer errores en el calculo del error porque, en casos
            particulares, el punto atraido puede quedar dentro de un prisma
            por la traslacion realizada para poder diferenciar

Nota 2:     No se controla internamente si los puntos atraidos estan situados
            en el interior de algun prisma

Original:   26-05-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Corregido error en el calculo de la gravedad del prisma que
              hacia que el signo saliese cambiado. Este error solo afectaba a
              la version de la funcion escrita en el lenguaje de GNU Octave,
              no a la escrita en C++
```

- **geo_GravedadPunto.m**:

```
Funcion:    geo_GravedadPunto.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula la componente vertical (eje Z) de la atraccion
            gravitatoria de un conjunto de puntos de masa sobre un conjunto de
```

```
            puntos dados
            La formulacion ha sido extraida de [HM-GEFI-1985], pag. 2

Entradas:   - Vector fila o matriz de tres columnas con las coordenadas (en
              metros) de los puntos atraidos en un sistema cartesiano
              dextrogiro. Cada fila de la matriz es un punto distinto:
              - Col. 1: Coordenada X del punto
              - Col. 2: Coordenada Y del punto
              - Col. 3: Coordenada Z del punto
            - Vector fila o matriz de tres columnas con las coordenadas (en
              metros) de los puntos atrayentes en un sistema cartesiano
              dextrogiro. Cada fila de la matriz es un punto distinto:
              - Col. 1: Coordenada X del punto
              - Col. 2: Coordenada Y del punto
              - Col. 3: Coordenada Z del punto
            - Escalar o vector columna que almacena la masa de cada punto
              atrayente. Si es un vector ha de tener el mismo numero de filas
              que la matriz de puntos atrayentes
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos atraidos. Dos posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2 Sz1^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos atrayentes. Dos
                posibilidades:
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Syi^2 Sz1^2];
                - Vector (o matriz) de seis columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxiyi Sxizi Syi^2 Syizi Szi^2];
              - Varianza de la masa de los puntos atrayentes (escalar o vector
                columna)
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Escalar o vector columna con la gravedad en cada punto debida a
              la atraccion de cada punto de masa. Si solo se ha introducido un
              punto atraido las filas del vector columna corresponden a la
              atraccion debida a cada punto de masa, mientras que si se han
              introducido varios puntos y varios puntos de masa el vector
              columna contiene la atraccion debida a cada punto de masa sobre
              cada punto:
              [Pm1P1,...,PmNP1,Pm1P2,...PmNP2,...,Pm1PM,...,PmNPM]^T
```

```
              donde Pm->Punto de masa (atrayente) y P->Punto
            - Estimacion de la varianza, Sg^2, de la gravedad calculada para
              cada punto. Escalar o vector columna con la misma estructura que
              el primer argumento de salida
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [grav]=geo_GravedadPunto(pAtraidos,pAtrayentes,masas);
            [grav]=geo_GravedadPunto(pAtraidos,pAtrayentes,masas,-1,-1,-1);
            [grav,varGrav]=geo_GravedadPunto(pAtraidos,pAtrayentes,masas,
            varCovPuntosAtraidos,varCovPuntosAtrayentes,varMasas);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m
                             - geo_AjustaFilasVectorMatriz.m
                             - geo_VectorMatrizSimInterna.m

Funciones relacionadas de GEO: - geo_GravedadCilindroVertical.m
                               - geo_GravedadEsfera.m
                               - geo_GravedadPrismaMacMillan.m
                               - geo_GravedadPrismaNagy.m
                               - geo_GravedadPrismaPuntos.m
                               - geo_GravedadRectangulo.m

Ultima revision: 12-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_GravedadRectangulo.m**:

```
Funcion:    geo_GravedadRectangulo.m

Proposito:  Calcula la componente vertical (eje Z) de la atraccion
            gravitatoria de un conjunto de rectangulos contenidos en el plano
            XZ sobre un conjunto de puntos (tambien en el plano XZ) dados en
            el exterior o sobre su superficie segun la formulacion de
            [BS-GCGI-1994]

Entradas:   - Vector fila o matriz de dos columnas con las coordenadas (en
              metros) de los puntos atraidos en un sistema cartesiano. Cada
              fila de la matriz es un punto distinto:
              - Col. 1: Coordenada X del punto
              - Col. 2: Coordenada Z del punto
            - Vector fila o matriz de cuatro columnas con las coordenadas (en
              metros) de los limites de las caras del rectangulo atrayente en
              un sistema cartesiano. Cada fila de la matriz es un rectangulo
              distinto:
              - Col. 1: Coordenada X1 del rectangulo| Con (x1,z1)<(x2,z2)
              - Col. 2: Coordenada X2 del rectangulo| respectivamente
              - Col. 3: Coordenada Z1 del rectangulo| Esta condicion no se
              - Col. 4: Coordenada Z2 del rectangulo| controla internamente
            - Escalar o vector columna que almacena la densidad de cada
              rectangulo (en kg/m^3). Si es un vector ha de tener el mismo
              numero de filas que la matriz de rectangulos
            Argumentos opcionales:
              - Vector o matriz que contiene informacion sobre las matrices
                varianza-covarianza de los puntos atraidos. Dos posibilidades:
                - Vector (o matriz) de dos columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto. Queda asi
```

```
                  expresada una matriz varianza-covarianza diagonal:
                  varCov=[Sxi^2 Sz1^2];
                - Vector (o matriz) de tres columnas, cada una de las cuales
                  contiene la varianza de cada coordenada del punto y sus
                  correlaciones. Queda asi expresada (por filas) una matriz
                  varianza-covarianza completa:
                  varCov=[Sxi^2 Sxizi Szi^2];
              - Vector (o matriz) de cuatro columnas, cada una de las cuales
                contiene la varianza de cada coordenada los limites de cada
                rectangulo:
                varCov=[Sx1^2 Sx2^2 Sz1^2 Sz2^2];
              - Varianza de la densidad (escalar o vector columna)
              Si, en vez de las matrices varianza-covarianza, se envian tres
              variables de valor -1, no se hace el calculo del error

Salidas:    - Escalar o vector columna con la gravedad en cada punto debida a
              la atraccion de cada rectangulo. Si solo se ha introducido un
              punto atraido las filas del vector columna corresponden a la
              atraccion debida a cada rectangulo, mientras que si se han
              introducido varios puntos y varios rectangulos el vector columna
              contiene la atraccion debida a cada rectangulo sobre cada punto:
              [Re1P1,...,ReNP1,Re1P2,...ReNP2,...,Re1PM,...,ReNPM]^T
              donde Re->Rectangulo y P->Punto y los rectangulos van de 1 a N y
              los puntos atraidos de 1 a M
            - Estimacion de la varianza, Sg^2, de la gravedad calculada para
              cada punto. Escalar o vector columna con la misma estructura que
              el primer argumento de salida
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [grav]=geo_GravedadRectangulo(ptos,rect,dens);
            [grav]=geo_GravedadRectangulo(ptos,rect,dens,-1,-1,-1);
            [grav,varGrav]=geo_GravedadRectangulo(ptos,rect,dens,varCovPtos,
            varRect,varDens);

Usa geoVarGlobal: Si

Nota 1:     Para realizar el calculo del error en esta funcion las derivadas
            parciales correspondientes se realizan de forma numerica. El
            elemento diferencial en coordenadas para calcular las citadas
            derivadas se calcula como factor del tamanyo del rectangulo
            atrayente. Este factor se almacena en la variable global:
              geoVarGlobal.diferencial.factDimCuerpPeq
            El diferencial en densidad tambien se almacena en la variable
            global:
              geoVarGlobal.diferencial.densidad
            Se hace notar la importancia que tiene introducir los valores de
            densidad en las mismas unidades que las de su diferencial (kg/m^3)
            para asegurar un correcto computo del error
            Debido al hecho de calcular las derivadas en forma numerica pueden
            aparecer errores en el calculo del error porque, en casos
            particulares, el punto atraido puede quedar dentro de un
            rectangulo por la traslacion realizada para poder diferenciar

Nota 2:     No se controla internamente si los puntos atraidos estan situados
            en el interior de algun rectangulo
```

```
Original:   26-05-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 15-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Reorganizacion de la ayuda
              Reescritura de la funcion para que no utilice la variable
              varargin como contenedor de los argumentos de entrada
              Reescritura completa de la funcion. Las versiones previas a esta
              fecha son erroreas
```

- **geo_GravedadSomigliana.m**:

```
Funcion:    geo_GravedadSomigliana.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula el valor de la gravedad normal sobre el elipsoide mediante
            la formula de Simigliana y una estimacion de su error (varianza)
            La formulacion ha sido extraida de [HM-GEFI-1985], pag. 70

Entradas:   - Latitud geodesica del punto de trabajo (escalar o vector
              columna), en radianes
            - Aceleracion de la gravedad en el ecuador para el elipsoide de
              trabajo, en m/s^2
            - Aceleracion de la gravedad en el polo para el elipsoide de
              trabajo, en m/s^2
            - Semieje mayor del elipsoide, en metros. En realidad no importa
              la unidad lineal que se utilice siempre y cuando sea la misma
              en las gravedades y el semieje del elipsoide.
            - Aplanamiento del elipsoide, (a-b)/a
            Argumento opcional:
              - Varianza de la latitud geodesica (escalar o vector columna),
                en radianes^2
              La varianza ha de ser pasada en valor absoluto (no se controla
              internamente)
              Si, en vez de la varianza, se envia una variable de valor -1, no
              se hace el calculo del error

Salidas:    - Gravedad normal en el punto de trabajo (escalar o vector
              columna), en la mismas unidades que las gravedades en el polo y
              el ecuador
            - Estimacion de la varianza, Sg^2, la gravedad normal calculada
              En el caso de no haber pasado datos para el calculo del error,
              este argumento de salida tiene el valor -1 (si se ha solicitado)

Ejemplo:    [gamma]=geo_GravedadSomigliana(latGeo,gammaA,gammaB,a,f);
            [gamma]=geo_GravedadSomigliana(latGeo,gammaA,gammaB,a,f,-1);
            [gamma,varGamma]=geo_GravedadSomigliana(latGeo,gammaA,gammaB,a,f,
            varLatGeo);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_HayQueCalcularErrores.m
                             - geo_ControlDominioLatGeoAF.m
                             - geo_ControlDominio.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 10-02-08
```

```
Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PrismaPuntoMasa.m**:

```
Funcion:    geo_PrismaPuntoMasa.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Pasa de un prisma a un punto en su centro conservando la masa

Entradas:   - Vector fila o matriz de seis columnas con las coordenadas de los
              limites de las caras del prisma en un sistema cartesiano
              dextrogiro. Cada fila de la matriz es un prisma distinto:
              - Col. 1: Coordenada X1 del prisma|
              - Col. 2: Coordenada X2 del prisma| Con (x1,y1,z1)<(x2,y2,z2)
              - Col. 3: Coordenada Y1 del prisma| respectivamente
              - Col. 4: Coordenada Y2 del prisma| Esta condicion no se
              - Col. 5: Coordenada Z1 del prisma| controla internamente
              - Col. 6: Coordenada Z2 del prisma|
            - Escalar o vector columna que almacena la densidad de cada prisma
            Argumentos opcionales:
              - Vector (o matriz) de seis columnas, cada una de las cuales
                contiene la varianza de cada coordenada los limites de cada
                prisma:
                varCov=[Sx1^2 Sx2^2 Sy1^2 Sy2^2 Sz1^2 Sz2^2];
              - Varianza de la densidad (escalar o vector columna)
              Las varianzas han de ser pasadas en valor absoluto (no se
              controla internamente)
              Si, en vez de las matrices varianza-covarianza, se envian dos
              variables de valor -1, no se hace el calculo del error

Salidas:    - Vector fila o matriz de tres columnas con las coordenadas de los
              puntos calculados. Cada fila de la matriz es un punto distinto:
              - Col. 1: Coordenada X del punto
              - Col. 2: Coordenada Y del punto
              - Col. 3: Coordenada Z del punto
            - Escalar o vector columna con la masa de cada punto calculado
            - Vector (o matriz) de tres columnas, cada una de las cuales
              contiene la varianza de cada coordenada del punto calculado
              Queda asi expresada una matriz varianza-covarianza diagonal:
              varCov=[Sxi^2 Syi^2 Sz1^2];
            - Estimacion de la varianza, Sm^2, de la masa calculada para cada
              punto. Escalar o vector columna
              En el caso de no haber pasado datos para el calculo del error,
              estos argumentos de salida tienen el valor -1 (si se han
              solicitado)

Ejemplo:    [punto,masa]=geo_PrismaPuntoMasa(prismas,densidades);
            [punto,masa]=geo_PrismaPuntoMasa(prismas,densidades,-1,-1);
            [punto,masa,varPunto,varMasa]=geo_PrismaPuntoMasa(prismas,
            densidades,varPrismas,varMasas);

Funciones utilizadas de GEO: - geo_HayQueCalcularErrores.m

Funciones relacionadas de GEO: - geo_GravedadCilindroVertical.m
                               - geo_GravedadEsfera.m
                               - geo_GravedadPrismaMacMillan.m
                               - geo_GravedadPrismaNagy.m
```

```
                                - geo_GravedadPrismaPuntos.m

Ultima revision: 12-02-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 18.3. Carpeta 03-probInvGrav

En esta carpeta se almacenan las funciones relacionadas con el problema inverso de la gravimetría.

- **geo_ParteZonaEnPrismas.m**:

```
Funcion:    geo_ParteZonaEnPrismas.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Divide una zona del espacio euclideo en prismas

Entradas:   - Vector fila de seis columnas con las coordenadas de los limites
              de la zona a dividir en prismas:
              - Col. 1: Coordenada X1 de la zona|
              - Col. 2: Coordenada X2 de la zona|
              - Col. 3: Coordenada Y1 de la zona| Con (x1,y1,z1)<(x2,y2,z2)
              - Col. 4: Coordenada Y2 de la zona| respectivamente
              - Col. 5: Coordenada Z1 de la zona|
              - Col. 6: Coordenada Z2 de la zona|
            - Vector fila de tres columnas con el paso de malla de las prismas
              en cada coordenada (en valor absoluto):
              - Col. 1: Paso de malla en X
              - Col. 2: Paso de malla en Y
              - Col. 3: Paso de malla en Z

Salidas:    - Matriz de 6 columnas donde cada fila es un prisma distinto:
              - Col. 1: Coordenada X1 del prisma|
              - Col. 2: Coordenada X2 del prisma|
              - Col. 3: Coordenada Y1 del prisma| Con (x1,y1,z1)<(x2,y2,z2)
              - Col. 4: Coordenada Y2 del prisma| respectivamente
              - Col. 5: Coordenada Z1 del prisma|
              - Col. 6: Coordenada Z2 del prisma|
            - Numero de prismas creados en la direccion X
            - Numero de prismas creados en la direccion Y
            - Numero de prismas creados en la direccion Z
            Los prismas de la particion de la zona se almacenan de oeste a
            este y de norte a sur en al plano y de cota superior a inferior en
            Z. Sirva de explicacion el siguiente ejemplo:

                                              1 -> [x11 x12 y11 y12 z11 z12
            ZS  1  2                          2 ->  x21 x22 y21 y22 z21 z22
                3  4                          3 ->  x31 x32 y31 y32 z31 z32
                ----  queda representada como 4 ->  x41 x42 y41 y42 z41 z42
            ZI  5  6                          5 ->  x51 x52 y51 y52 z51 z52
                7  8                          6 ->  x61 x62 y61 y62 z61 z62
                                              7 ->  x71 x72 y71 y72 z71 z72
                                              8 ->  x81 x82 y81 y82 z81 z82]
            ZS -> Z superior
            ZI -> Z inferior
```

```
            Si el paso de malla no es congruente con los limites impuestos se
            ajusta para que lo sea al paso mas cercano. Se lanza un warning
            avisando de la situacion

Ejemplo:    [prismas,nX,nY,nZ]=geo_ParteZonaEnPrismas(zona,pasos);

Funciones utilizadas de GEO: - geo_EsNatural.m

Funciones relacionadas de GEO: - geo_CreaLineaPuntosElipsoide.m
                               - geo_CreaLineaPuntosPlano.m
                               - geo_CreaMallaPuntosElipsoide.m
                               - geo_CreaMallaPuntosPlano.m

Ultima revision: 06-04-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

## 18.4. Carpeta 04-obsGravedad

En esta carpeta se almacenan las funciones relacionadas con la reducción de observaciones de gravedad.

- **geo_CalcLectDialRectaLRG.m**:

```
Funcion:    geo_CalcLectDialRectaLRG.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Calcula las posiciones en un vector columna de todos los elementos
            iguales entre si que esten situados de forma consecutiva

Entradas:   - Un vector columna

Salidas:    - Una matriz de tamano Nx3 que contiene el elemento del vector
              original y las posiciones en que comienza y termina un grupo de
              elementos iguales consecutivos de ese elemento. Su estructura es
              la siguiente:
              posiciones=[dato1  filInicio1  filFin1
                          ........................
                          datoN  filInicioN  filFinN];
              Cada fila corresponde a un grupo de elementos iguales:
              - Col. 1: Dato almacenado en el vector original
              - Col. 2: Fila del vector original en la que se encuentra el
                        dato de trabajo
              - Col. 3: Fila del vector original en la que termina una serie
                        de datos iguales

Ejemplo:    [posiciones]=geo_PosGruposVector(vector);

Funciones utilizadas de GEO: - geo_EsEscalar.m
                             - geo_DesactivaControles.m
                             - geo_MinCuadEcObs.m
                             - geo_RestauraControles.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 15-08-08
```

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

# Capítulo 19
# Sección 11-astronomia

En esta carpeta se almacenan las funciones relacionadas con la astronomía de posición.

## 19.1. Carpeta 01-tiempo

En esta carpeta se almacenan las funciones relacionadas con el procesamiento de datos de tiempo.

- **geo_EsBisiesto.m**:

```
Funcion:    geo_EsBisiesto.m

Proposito:  Comprueba si un anyo es bisiesto o no

Entradas:   - Un anyo, numero entero, escalar o vector columna

Salidas:    - Si el argumento de entrada es un escalar:
              - 1, si es un anyo bisiesto
              - 0, en caso contrario
            - Si el argumento de entrada es un vector columna:
              - Un vector columna del mismo tamanyo que el vector de entrada
                que contiene en cada posicion correspondiente al vector de
                entrada:
                - 1, si en la posicion correspondiente hay un anyo bisiesto
                - 0, en caso contrario

Ejemplo:    bisiesto=geo_EsBisiesto(anyo);

Usa geoVarGlobal: Si

Original:   25-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geo_FechaDiaJuliano.m**:

```
Funcion:    geo_FechaDiaJuliano.m

Proposito:  Calcula el dia juliano para una fecha dada

Entradas:   - Hora del dia, en formato decimal (escalar o vector columna)
            - Dia del mes (escalar o vector columna)
            - Mes del anyo (escalar o vector columna)
            - Anyo (escalar o vector columna)
```

```
Salidas:    - Dia juliano correspondiente (escalar o vector columna)

Ejemplo:    diaJuliano=geo_FechaDiaJuliano(hora,dia,mes,anyo);

Usa geoVarGlobal: Si

Nota 1:     Esta rutina es valida para cualquier fecha desde el anyo -4713

Nota 2:     Esta funcion es una copia (adapdata a la sintaxis de GNU Octave)
            de la subrutina 'ETJULN' del programa 'PREDICT.FOR' del paquete de
            analisis de mareas terrestres 'ETERNA 3.4'

Nota 3:     La subrutina 'ETJULN' es una modificacion de la subrutina 'MJD'
            del libro [MO-ASPC-1989]

Original:   03-03-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

## 19.2. Carpeta 02-gps

En esta carpeta se almacenan las funciones relacionadas con el procesamiento de datos GPS.

## 19.3. Carpeta 03-efemerides

En esta carpeta se almacenan las funciones relacionadas con el cálculo de efemérides.

- **geo_EfemeridesSimon.m**:

```
Funcion:    geo_EfemeridesSimon.m

Proposito:  Calcula los elementos astronomicos segun la formulacion que se
            puede encontrar en [SI-NEPF-1994] para una fecha dada

Entradas:   - Fecha juliana modificada (dÃŋas julianos-2400000.5) en tiempo
              dinamico baricentrico (TDB), (escalar o vector columna)

Salidas:    - Vector fila o matriz con los elementos astronomicos calculados
              (todos los datos en radianes). Cada columna es:
              - Col.  1: Tiempo medio local lunar
              - Col.  2: Longitud media de la Luna
              - Col.  3: Longitud media del Sol
              - Col.  4: Longitud media del perigeo de la orbita de la Luna
              - Col.  5: Longitud negativa media del nodo ascendente de la
                         orbita de la Luna
              - Col.  6: Longitud media del perigeo de la orbita del Sol
              - Col.  7: Longitud media de Mercurio
              - Col.  8: Longitud media de Venus
              - Col.  9: Longitud media de Marte
              - Col. 10: Longitud media de Jupiter
              - Col. 11: Longitud media de Saturno
            - Vector fila o matriz con las derivadas con respecto al tiempo de
              los elementos astronomicos calculados (todos los datos en
              radianes/hora). Cada columna contiene la derivada
```

```
              correspondiente al elemento astronomico que ocupa la misma
              columna

Ejemplo:    [elementos,derivadas]=geo_EfemeridesSimon(fecha);

Usa geoVarGlobal: No

Nota 1:     Esta funcion es una copia (adapdata a la sintaxis de GNU Octave)
            de la subrutina 'ETASTN' del programa 'PREDICT.FOR' del paquete de
            analisis de mareas terrestres 'ETERNA 3.4'

Nota 2:     Esta funcion puede dar resultados ligeramente diferentes (a partir
            del octavo o noveno decimal) de los obtenidos con su equivalente
            programada en C o Fortran

Original:   05-03-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

# Capítulo 20
# Sección 12-instrumento

En esta carpeta se guardan funciones relacionadas con el manejo de instrumentos topográficos.

## 20.1. Carpeta 01-estructuraInstrumento

Esta carpeta almacena las funciones relacionadas con la creación de la estructura `instrumento`.

## 20.2. Carpeta 02-calculosInstrumento

En esta carpeta se guardan las funciones relacionadas con el calculo de errores a priori de cada instrumento topográfico.

# Capítulo 21
# Sección 13-datos

En esta carpeta se almacenan los ficheros de datos necesarios para la ejecución de ciertas funciones de GEO.

- **bibliografia.dat**:

```
Fichero:    bibliografia.dat

Proposito:  Almacena las definiciones de la referencias bibliograficas
            utilizadas para la programacion de las funciones de GEO

Formato:    Para cada referencia se definen:
            - Linea 1 (L1):
              - Col. 1: Comentario acerca del parametro almacenado. Cadena de
                        texto de 15 caracteres justificada a la derecha
              - Col. 2: Identificador del tipo de documento almacenado. Cadena
                        de texto de 15 caracteres justificada a la derecha
                        Dos posibilidades (cadenas de texto):
                          - 'ARTICULO': Si el documento almacenado es un
                                        articulo
                          - 'LIBRO': Si el documento almacenado es un libro
            - Linea 2 (L2):
              - Col. 1: Comentario acerca del parametro almacenado. Cadena de
                        texto de 15 caracteres justificada a la derecha
              - Col. 2: Identificador del documento almacenado. Cadena de
                        texto de 30 caracteres justificada a la derecha
            - Linea 3 (L3):
              - Col. 1: Comentario acerca del parametro almacenado. Cadena de
                        texto de 15 caracteres justificada a la derecha
              - Col. 2: Numero de lineas (N1) a partir de la actual en las que
                        se almacena el titulo del documento almacenado. Numero
                        entero >=0, como cadena de texto de 5 caracteres
                        justificada a la derecha
            - Lineas L3+1 a L3+N1:
              - Titulo del documento almacenado. Cadenas de texto de 80
                caracteres justificadas a la derecha
            - Linea L3+N1+1 (L4):
              - Col. 1: Comentario acerca del parametro almacenado. Cadena de
                        texto de 15 caracteres justificada a la derecha
              - Col. 2: Numero de lineas (N2) a partir de la actual en las que
                        se almacena el nombre de la revista o el de la
                        editorial a la que pertenece el documento almacenado
                        Numero entero >=0, como cadena de texto de 5
                        caracteres justificada a la derecha
            - Lineas L4+1 a L4+N2:
```

```
                - Nombre de la revista o de la editorial a la que pertene el
                  documento almacenado. Cadenas de texto de 80 caracteres
                  justificadas a la derecha
              - Linea L4+N2+1 (L5):
                - Col. 1: Comentario acerca del parametro almacenado. Cadena de
                          texto de 15 caracteres justificada a la derecha
                - Col. 2: Numero de lineas (N3) a partir de la actual en las que
                          se almacena una direccion web donde encontrar
                          informacion acerca del documento almacenado. Numero
                          entero >=0, como cadena de texto de 5 caracteres
                          justificada a la derecha
              - Lineas L5+1 a L5+N3:
                - Direccion web donde encontrar informacion acerca del documento
                  almacenado. Cadenas de texto de 80 caracteres justificadas a
                  la derecha
              - Linea L5+N3+1 (L6):
                - Col. 1: Comentario acerca del parametro almacenado. Cadena de
                          texto de 15 caracteres justificada a la derecha
                - Col. 2: Numero de lineas (N4) a partir de la actual en las que
                          se almacena informacion adicional acerca del documento
                          Numero entero >=0, como cadena de texto de 5
                          caracteres justificada a la derecha
              - Lineas L6+1 a L6+N4:
                - Informacion adicional acerca del documento almacenado. Cadenas
                  de texto de 80 caracteres justificadas a la derecha

Nota:       Entre cada bloque de definicion de referencia puede haber un
            numero arbitrario de lineas en blanco o de comentarios

Original:   26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 12-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Cambio en las longitudes de la segunda columna de la lineas
              primera y segunda de cada referencia
```

- **datum.dat**:

```
Fichero:    datum.dat

Proposito:  Almacena las definiciones de los datum utilizados en GEO

Formato:    Para cada datum se definen:

Nota:       FICHERO POR DESARROLLAR

Original:   26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **elipsoide.dat**:

```
Fichero:    elipsoide.dat

Proposito:  Almacena las definiciones de los elipsoides utilizados en GEO. Se
            incluyen todos los elipsoides definidos en GeoTrans (v.2.2.3)

Formato:    Para cada elipsoide se definen las siguientes lineas:
```

```
- La primera columna siempre es un comentario acerca del parametro
  almacenado en la linea. Es una cadena de texto de 30 caracteres,
  justificada a la derecha
- Linea 1:
  - Col. 2: Codigo identificador del elipsoide. Cadena de texto de
            15 caracteres justificada a la derecha
- Linea 2:
  - Col. 2: Semieje mayor del elipsoide, en metros. Real en doble
            precision como cadena de texto de 25 caracteres
            justificada a la derecha
- Linea 3:
  - Col. 2: Inverso del aplanamiento del elipsoide, adimensional
            Real en doble precision como cadena de texto de 25
            caracteres justificada a la derecha
- Linea 4:
  - Col. 2: Numero identificador de existencia de parametros
            dinamicos del elipsoide. Dos posibilidades:
            - 0: No hay parametros dinamicos
            - 1: Si hay parametros dinamicos
            Cadena de texto de 25 caracteres justificada a la
            derecha. Todavia no hay soporte para estos parametros,
            por lo que este valor siempre ha de ser 0
- Linea 5:
  - Col. 2: Numero de lineas (N) a partir de la actual en las que
            se almacena informacion adicional acerca del elipsoide
            Numero entero >=0. Cadena de texto de 25 caracteres
            justificada a la derecha
- Lineas 6 a 6+N-1:
  - Informacion adicional acerca del elipsoide almacenado. Cadenas
    de texto de 80 caracteres justificadas a la derecha
```

```
Nota 1:     Entre cada bloque de definicion de modelo de geoide puede haber un
            numero arbitrario de lineas en blanco o de comentarios

Nota 2:     No se hace distincion entre mayusculas y minusculas

Original:   13-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **geoide.dat**:

```
Fichero:    geoide.dat

Proposito:  Almacena las definiciones de los geoides utilizados en GEO

Formato:    Para cada geoide se definen las siguientes lineas:
            - La primera columna siempre es un comentario acerca del parametro
              almacenado en la linea. Es una cadena de texto de 30 caracteres,
              justificada a la derecha
            - Linea 1:
              - Col. 2: Codigo identificador del geoide. Cadena de texto de 20
                        caracteres justificada a la derecha
            - Linea 2:
              - Col. 2: Codigo identificador del elipsoide al que esta
                        referido el geoide. Cadena de texto de 25 caracteres
                        justificada a la derecha
            - Linea 3:
```

```
  - Col. 2: Numero identificador del sistema de coordenadas en que
            vienen expresados los puntos dato. Dos posibilidades:
            - 0: Coordenadas cartesianas
            - 1: Coordenadas geodesicas
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 4:
  - Col. 2: Numero identificador de cobertura global en la
            direccion norte-sur. Dos posibilidades:
            - 0: No hay cobertura global
            - 1: Si hay cobertura global
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 5:
  - Col. 2: Numero identificador de cobertura global en la
            direccion este-oeste. Dos posibilidades:
            - 0: No hay cobertura global
            - 1: Si hay cobertura global
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 6:
  - Col. 2: Numero identificador de la repeticion del primer punto
            en la direccion este-oeste de cada fila de la malla de
            datos. Dos posibilidades:
            - 0: No se repite el primer punto
            - 1: Si se repite el primer punto
            Este argumento solo se tendra en cuenta si el modelo
            tiene cobertura global en la direccion este-oeste
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 7:
  - Col. 2: Numero identificador de existencia de informacion de
            varianza de los datos. Dos posibilidades:
            - 0: No hay informacion de varianza
            - 1: Si hay informacion de varianza
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 8:
  - Col. 2: Nombre del tipo de dato de GEO utilizado para
            almacenar los datos de ondulacion en el fichero
            binario correspondiente. Los nombres del tipo de dato
            se encuentran definidos en la estructura
            global geoVarGlobal.nombreDatoBinGeo{}. Cadena de
            texto de 25 caracteres justificada a la derecha
- Linea 9:
  - Col. 2: Numero identificador del tipo de unidades en las que
            estan almacenados los datos de ondulacion. Los numeros
            identificadores de tipo de dato se encuentran
            definidos en la estructura global
            geoVarGlobal.unidades{}. Cadena de texto de 25
            caracteres justificada a la derecha
- Linea 10:
  - Col. 2: Nombre del tipo de almacenamiento de los datos de
            ondulacion en el fichero binario correspondiente. Los
            nombres identificadores del tipo de almacenamiento se
            encuentran definidos en la estructura global
            geoVarGlobal.tipoAlmacenBinarioGeo{}. Cadena de texto
```

```
            de 25 caracteres justificada a la derecha
- Linea 11:
  - Col. 2: Nombre del tipo de dato de GEO utilizado para
            almacenar los datos de varianza en el fichero binario
            correspondiente. Los nombres del tipo de dato se
            encuentran definidos en la estructura
            global geoVarGlobal.nombreDatoBinGeo{}. Cadena de
            texto de 25 caracteres justificada a la derecha
            Aunque el modelo de geoide no contenga datos de
            varianza, en esta linea se ha de guardar un parametro
            valido
- Linea 12:
  - Col. 2: Numero identificador del tipo de unidades en las que
            estan almacenados los datos de varianza. Los numeros
            identificadores de tipo de dato se encuentran
            definidos en la estructura global
            geoVarGlobal.unidades{}. Cadena de texto de 25
            caracteres justificada a la derecha. Aunque el modelo
            de geoide no contenga datos de varianza, en esta linea
            se ha de guardar un parametro valido
- Linea 13:
  - Col. 2: Nombre del tipo de almacenamiento de los datos de
            varianza en el fichero binario correspondiente. Los
            nombres identificadores del tipo de almacenamiento se
            encuentran definidos en la estructura global
            geoVarGlobal.tipoAlmacenBinarioGeo{}. Cadena de texto
            de 25 caracteres justificada a la derecha. Aunque el
            modelo de geoide no contenga datos de varianza, en
            esta linea se ha de guardar un parametro valido
- Linea 14:
  - Col. 2: Numero identificador de existencia de puntos sin dato
            de ondulacion. Dos posibilidades:
            - 0: No hay puntos sin dato
            - 1: Si hay puntos sin dato
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 15:
  - Col. 2: Valor indicador (numero entero) de punto sin dato de
            ondulacion. Cadena de texto de 25 caracteres
            justificada a la derecha. Aunque el modelo de geoide
            no contenga datos vacios, en esta linea se ha de
            guardar un parametro valido
- Linea 16:
  - Col. 2: Numero identificador de existencia de puntos sin dato
            de varianza. Dos posibilidades:
            - 0: No hay puntos sin dato
            - 1: Si hay puntos sin dato
            Cadena de texto de 25 caracteres justificada a la
            derecha. Aunque el modelo de geoide no contenga datos
            de varianza vacios, en esta linea se ha de guardar un
            parametro valido
- Linea 17:
  - Col. 2: Valor indicador (numero entero) de punto sin dato de
            varianza. Cadena de texto de 25 caracteres justificada
            a la derecha. Aunque el modelo de geoide no contenga
            datos de varianza vacios, en esta linea se ha de
            guardar un parametro valido
```

```
- Linea 18:
  - Col. 2: Limite norte de la malla de datos. Dos posibilidades:
            - Si el sistema de coordenadas es cartesiano, maximo
              valor de la coordenada Y
            - Si el sistema de coordenadas es geodesico, latitud
              en graduacion seudo sexagesimal
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 19:
  - Col. 2: Limite sur de la malla de datos. Dos posibilidades:
            - Si el sistema de coordenadas es cartesiano, minimo
              valor de la coordenada Y
            - Si el sistema de coordenadas es geodesico, latitud
              en graduacion seudo sexagesimal
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 20:
  - Col. 2: Limite este de la malla de datos. Dos posibilidades:
            - Si el sistema de coordenadas es cartesiano, maximo
              valor de la coordenada X
            - Si el sistema de coordenadas es geodesico, longitud
              en graduacion seudo sexagesimal
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 21:
  - Col. 2: Limite oeste de la malla de datos. Dos posibilidades:
            - Si el sistema de coordenadas es cartesiano, minimo
              valor de la coordenada X
            - Si el sistema de coordenadas es geodesico, longitud
              en graduacion seudo sexagesimal
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 22:
  - Col. 2: Espaciado (paso de malla) entre los puntos dato (en
            valor absoluto) en la direccion norte-sur (si el
            sistema de coordenadas es geodesico, en graduacion
            seudo sexagesimal). Cadena de texto de 25 caracteres
            justificada a la derecha
- Linea 23:
  - Col. 2: Espaciado (paso de malla) entre los puntos dato (en
            valor absoluto) en la direccion este-oeste (si el
            sistema de coordenadas es geodesico, en graduacion
            seudo sexagesimal). Cadena de texto de 25 caracteres
            justificada a la derecha
- Linea 24:
  - Col. 2: Numero de lineas (N) a partir de la actual en las que
            se almacena informacion adicional acerca del modelo de
            geoide. Numero entero >=0. Cadena de texto de 25
            caracteres justificada a la derecha
- Lineas 25 a 25+N-1:
  - Informacion adicional acerca del modelo de geoide almacenado
    Cadenas de texto de 80 caracteres justificadas a la derecha

Nota 1:    Entre cada bloque de definicion de modelo de geoide puede haber un
           numero arbitrario de lineas en blanco o de comentarios

Nota 2:    No se hace distincion entre mayusculas y minusculas
```

```
Nota 3:     Los limites este y oeste se determinan de la siguiente forma:
            - Si el sistema de coordenadas es geodesico:
              Imaginemos la malla de puntos proyectada sobre un mapa de
              Mercator. Observando el mapa en su punto medio en longitud, el
              limite este sera la longitud de los puntos extremos a la derecha
              y el oeste la de los puntos extremos a la izquierda. Si el
              modelo tiene cobertura global, el punto observado sera el
              antimeridiano del meridiano de solape de la malla. En este
              ultimo caso caso, los limites en longitud escritos en el fichero
              han de coincidir en valor absoluto
              Las latitudes se expresaran en el dominio [-90 90] y las
              longitudes en el dominio ]-180 0] U [0 180]
            - Si el sistema de coordenadas es cartesiano:
              Los limites este y oeste corresponden a las coordenadas X maxima
              y minima, respectivamente

Nota 4:     Los datos de ondulacion se almacenan en ficheros en formato
            binario. La sucesion de valores en el fichero correspondiente
            responde a una ordenacion de oeste a este y de norte a sur de los
            puntos. Por ejemplo:

            Una malla como la siguiente:

                                    N

                                | 1 2 3 |
                              W | 4 5 6 | E
                                | 7 8 9 |

                                    S

            Dara lugar a una sucesion en el fichero binario como esta:

            FICHERO_WENS = 123456789

            Donde cada valor ocupara el espacio correspondiente al dato
            indicado en la definicion del modelo

Original:   06-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **instrumento.dat**:

```
Fichero:    instrumento.dat

Proposito:  Almacena las definiciones de los instrumentos de observacion
            utilizados en GEO

Formato:    Para cada instrumento se definen:

Nota:       FICHERO POR DESARROLLAR

Original:   26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **mde.dat**:

```
Fichero:    mde.dat

Proposito:  Almacena las definiciones de los modelos digitales de elevacion
            (MDE) utilizados en GEO

Formato:    Para cada MDE se definen las siguientes lineas:
            - La primera columna siempre es un comentario acerca del parametro
              almacenado en la linea. Es una cadena de texto de 30 caracteres,
              justificada a la derecha
            - Linea 1:
              - Col. 2: Codigo identificador del MDE. Cadena de texto de 20
                        caracteres justificada a la derecha
            - Linea 2:
              - Col. 2: Numero identificador del tipo de altitudes que
                        almacena el MDE. Dos posibilidades:
                        - 0: Altitud elipsoidal
                        - 1: Altitud ortometrica
                        Cadena de texto de 25 caracteres justificada a la
                        derecha
            - Linea 3:
              - Col. 2: Codigo identificador de la superficie de referencia a
                        la que se refieren las altitudes. Dos posibilidades:
                        - Si las altitudes son elipsoidales, codigo del
                          elipsoide al que vienen referidas
                        - Si las altitudes son ortometricas, codigo del geoide
                          al que vienen referidas
                        Cadena de texto de 25 caracteres justificada a la
                        derecha
            - Linea 4:
              - Col. 2: Codigo identificador del elipsoide al que estan
                        referidas las coordenadas planimetricas de los puntos
                        dato. Si se almacenan altitudes elipsoidades, este
                        parametro ha de ser igual al almacenado en la linea
                        correspondiente al codigo identificador de la
                        superficie de referencia a la que se refieren las
                        altitudes. Cadena de texto de 25 caracteres
                        justificada a la derecha.
            - Linea 5:
              - Col. 2: Numero identificador del sistema de coordenadas en que
                        vienen expresados los puntos dato. Dos posibilidades:
                        - 0: Coordenadas cartesianas
                        - 1: Coordenadas geodesicas
                        Cadena de texto de 25 caracteres justificada a la
                        derecha
            - Linea 6:
              - Col. 2: Numero identificador de cobertura global en la
                        direccion norte-sur. Dos posibilidades:
                        - 0: No hay cobertura global
                        - 1: Si hay cobertura global
                        Cadena de texto de 25 caracteres justificada a la
                        derecha
            - Linea 7:
              - Col. 2: Numero identificador de cobertura global en la
                        direccion este-oeste. Dos posibilidades:
                        - 0: No hay cobertura global
                        - 1: Si hay cobertura global
```

```
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 8:
  - Col. 2: Numero identificador de la repeticion del primer punto
            en la direccion este-oeste de cada fila de la malla de
            datos. Dos posibilidades:
            - 0: No se repite el primer punto
            - 1: Si se repite el primer punto
            Este argumento solo se tendra en cuenta si el modelo
            tiene cobertura global en la direccion este-oeste
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 9:
  - Col. 2: Numero identificador de existencia de informacion de
            varianza de los datos. Dos posibilidades:
            - 0: No hay informacion de varianza
            - 1: Si hay informacion de varianza
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 10:
  - Col. 2: Nombre del tipo de dato de GEO utilizado para
            almacenar los datos de altitud en el fichero binario
            correspondiente. Los nombres del tipo de dato se
            encuentran definidos en la estructura
            global geoVarGlobal.nombreDatoBinGeo{}. Cadena de
            texto de 25 caracteres justificada a la derecha
- Linea 11:
  - Col. 2: Numero identificador del tipo de unidades en las que
            estan almacenados los datos de altitud. Los numeros
            identificadores de tipo de dato se encuentran
            definidos en la estructura global
            geoVarGlobal.unidades{}. Cadena de texto de 25
            caracteres justificada a la derecha
- Linea 12:
  - Col. 2: Nombre del tipo de almacenamiento de los datos de
            altitud en el fichero binario correspondiente. Los
            nombres identificadores del tipo de almacenamiento se
            encuentran definidos en la estructura global
            geoVarGlobal.tipoAlmacenBinarioGeo{}. Cadena de texto
            de 25 caracteres justificada a la derecha
- Linea 13:
  - Col. 2: Nombre del tipo de dato de GEO utilizado para
            almacenar los datos de varianza en el fichero binario
            correspondiente. Los nombres del tipo de dato se
            encuentran definidos en la estructura
            global geoVarGlobal.nombreDatoBinGeo{}. Cadena de
            texto de 25 caracteres justificada a la derecha
            Aunque el MDE no contenga datos de varianza, en esta
            linea se ha de guardar un parametro valido
- Linea 14:
  - Col. 2: Numero identificador del tipo de unidades en las que
            estan almacenados los datos de varianza. Los numeros
            identificadores de tipo de dato se encuentran
            definidos en la estructura global
            geoVarGlobal.unidades{}. Cadena de texto de 25
            caracteres justificada a la derecha. Aunque el MDE
            no contenga datos de varianza, en esta linea se ha de
```

```
            guardar un parametro valido
- Linea 15:
  - Col. 2: Nombre del tipo de almacenamiento de los datos de
            varianza en el fichero binario correspondiente. Los
            nombres identificadores del tipo de almacenamiento se
            encuentran definidos en la estructura global
            geoVarGlobal.tipoAlmacenBinarioGeo{}. Cadena de texto
            de 25 caracteres justificada a la derecha. Aunque el
            MDE no contenga datos de varianza, en esta linea se ha
            de guardar un parametro valido
- Linea 16:
  - Col. 2: Numero identificador de existencia de puntos sin dato
            de altitud. Dos posibilidades:
            - 0: No hay puntos sin dato
            - 1: Si hay puntos sin dato
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 17:
  - Col. 2: Valor indicador (numero entero) de punto sin dato de
            altitud. Cadena de texto de 25 caracteres justificada
            a la derecha. Aunque el MDE no contenga datos vacios,
            en esta linea se ha de guardar un parametro valido
- Linea 18:
  - Col. 2: Numero identificador de existencia de puntos sin dato
            de varianza. Dos posibilidades:
            - 0: No hay puntos sin dato
            - 1: Si hay puntos sin dato
            Cadena de texto de 25 caracteres justificada a la
            derecha. Aunque el MDE no contenga datos de varianza
            vacios, en esta linea se ha de guardar un parametro
            valido
- Linea 19:
  - Col. 2: Valor indicador (numero entero) de punto sin dato de
            varianza. Cadena de texto de 25 caracteres justificada
            a la derecha. Aunque el MDE no contenga datos de
            varianza vacios, en esta linea se ha de guardar un
            parametro valido
- Linea 20:
  - Col. 2: Limite norte de la malla de datos. Dos posibilidades:
            - Si el sistema de coordenadas es cartesiano, maximo
              valor de la coordenada Y
            - Si el sistema de coordenadas es geodesico, latitud
              en graduacion seudo sexagesimal
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 21:
  - Col. 2: Limite sur de la malla de datos. Dos posibilidades:
            - Si el sistema de coordenadas es cartesiano, minimo
              valor de la coordenada Y
            - Si el sistema de coordenadas es geodesico, latitud
              en graduacion seudo sexagesimal
            Cadena de texto de 25 caracteres justificada a la
            derecha
- Linea 22:
  - Col. 2: Limite este de la malla de datos. Dos posibilidades:
            - Si el sistema de coordenadas es cartesiano, maximo
              valor de la coordenada X
```

```
                - Si el sistema de coordenadas es geodesico, longitud
                  en graduacion seudo sexagesimal
                Cadena de texto de 25 caracteres justificada a la
                derecha
        - Linea 23:
          - Col. 2: Limite oeste de la malla de datos. Dos posibilidades:
                - Si el sistema de coordenadas es cartesiano, minimo
                  valor de la coordenada X
                - Si el sistema de coordenadas es geodesico, longitud
                  en graduacion seudo sexagesimal
                Cadena de texto de 25 caracteres justificada a la
                derecha
        - Linea 24:
          - Col. 2: Espaciado (paso de malla) entre los puntos dato (en
                valor absoluto) en la direccion norte-sur (si el
                sistema de coordenadas es geodesico, en graduacion
                seudo sexagesimal). Cadena de texto de 25 caracteres
                justificada a la derecha
        - Linea 25:
          - Col. 2: Espaciado (paso de malla) entre los puntos dato (en
                valor absoluto) en la direccion este-oeste (si el
                sistema de coordenadas es geodesico, en graduacion
                seudo sexagesimal). Cadena de texto de 25 caracteres
                justificada a la derecha
        - Linea 26:
          - Col. 2: Numero de lineas (N) a partir de la actual en las que
                se almacena informacion adicional acerca del MDE
                Numero entero >=0. Cadena de texto de 25 caracteres
                justificada a la derecha
        - Lineas 27 a 27+N-1:
          - Informacion adicional acerca del MDE almacenado. Cadenas de
            texto de 80 caracteres justificadas a la derecha

Nota 1:  Entre cada bloque de definicion de MDE puede haber un numero
         arbitrario de lineas en blanco o de comentarios

Nota 2:  No se hace distincion entre mayusculas y minusculas

Nota 3:  Los limites este y oeste se determinan de la siguiente forma:
         - Si el sistema de coordenadas es geodesico:
           Imaginemos la malla de puntos proyectada sobre un mapa de
           Mercator. Observando el mapa en su punto medio en longitud, el
           limite este sera la longitud de los puntos extremos a la derecha
           y el oeste la de los puntos extremos a la izquierda. Si el
           modelo tiene cobertura global, el punto observado sera el
           antimeridiano del meridiano de solape de la malla. En este
           ultimo caso caso, los limites en longitud escritos en el fichero
           han de coincidir en valor absoluto
           Las latitudes se expresaran en el dominio [-90 90] y las
           longitudes en el dominio ]-180 0] U [0 180]
         - Si el sistema de coordenadas es cartesiano:
           Los limites este y oeste corresponden a las coordenadas X maxima
           y minima, respectivamente

Nota 4:  Los datos de altitud se almacenan en ficheros en formato binario
         La sucesion de valores en el fichero correspondiente responde a
         una ordenacion de oeste a este y de norte a sur de los puntos. Por
```

```
ejemplo:

Una malla como la siguiente:

                          N

                      | 1 2 3 |
                  W   | 4 5 6 |   E
                      | 7 8 9 |

                          S

Dara lugar a una sucesion en el fichero binario como esta:

FICHERO_WENS = 123456789

Donde cada valor ocupara el espacio correspondiente al dato
indicado en la definicion del modelo

Original:   09-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **proyCarto.dat**:

```
Fichero:    proyCarto.dat

Proposito:  Almacena las definiciones de las proyecciones cartograficas
            utilizadas en GEO

Formato:    Para cada proyeccion se definen:

Nota:       FICHERO POR DESARROLLAR

Original:   26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

- **transfDatum.dat**:

```
Fichero:    transfDatum.dat

Proposito:  Almacena las definiciones de las transformaciones entre sistemas
            de referencia utilizadas en GEO

Formato:    Para cada transformacion se definen:

Nota:       FICHERO POR DESARROLLAR

Original:   26-12-2008, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: Ninguna
```

# Capítulo 22
# Sección 14-pruebas

En esta carpeta se almacenan los programas de prueba de las funciones de GEO.

- **geo_PruebaCalcOndulaciones.m**:

- **geo_PruebaCalcRedGps.m**:

```
Funcion:    geo_PruebaCalcRedGps.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba la correcta programacion del calculo de coordenadas
            aproximadas en una red GPS

Entradas:   Ninguna

Salidas:    - Un fichero en el directorio de trabajo

Ejemplo:    geo_PruebaCalcRedGps;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_CalculoRedGps.m
                             - geo_ImprimeTiempo.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 15-11-06

Nota: Problema sacado del libro [WO-ADCO-1997], pags. 309 y ss. Se han
      cambiado los nombres de los puntos que aparecen en el libro por letras
      de la forma:
      A -> 1000
      B -> 2000
      C -> 3000
      D -> 4000
      E -> 5000
      F -> 6000

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaComparaGeoides.m**:

- **geo_PruebaCurvado.m**:

```
Funcion:    geo_PruebaCurvado.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba la correcta programacion de las funciones de
            triangulacion de Delaunay y generacion de curvas de nivel

Entradas:   Ninguna

Salidas:    - Un fichero DXF en el directorio de trabajo
            - Un fichero en el directorio de trabajo que contiene un listado
              de los puntos utilizados en formato 'Codigo X Y Z'

Ejemplo:    geo_PruebaCurvado;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_Elipsoide.m
                             - geo_Mde.m
                             - geo_Geoide.m
                             - geo_CreaMallaPuntosElipsoide.m
                             - geo_GmsRad.m
                             - geo_AltitudMdeMalla.m
                             - geo_OndGeoideMalla.m
                             - geo_GeoTranMerc.m
                             - geo_Delaunay.m
                             - geo_CurvadoTin.m
                             - geo_DibujaTin.m
                             - geo_DibujaCurvado.m
                             - geo_Dibujo.m
                             - geo_Lista.m
                             - geo_CreaCapaDibujo.m
                             - geo_DameElementoLista.m
                             - geo_CreaPuntoDibujo.m
                             - geo_CreaPolilineaDibujo.m
                             - geo_LimitesVisualDibujo.m
                             - geo_CreaDibujoDxf.m
                             - geo_ImprimeTiempo.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 29-08-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaDibujoGeomadrid.m**:

- **geo_PruebaDosDist.m**:

- **geo_PruebaDxf.m**:

```
Funcion:    geo_PruebaDxf.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba la correcta programacion de las funciones de dibujo en
            formato DXF

Entradas:   Ninguna

Salidas:    - Un fichero DXF en el directorio de trabajo

Ejemplo:    geo_PruebaDxf;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_ImprimeTiempo.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 29-04-07

Nota: Red mixta de Robledo de Chavela. Dimensiones de simbolos y textos para
      impresion a escala 1/50000. Para las elipses de error 1cm en el plano
      corresponde a 2cm en la realidad a escala 1/50000

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaEspacio.m**:

- **geo_PruebaEspacioError.m**:

- **geo_PruebaEspacioErrorDef.m**:

- **geo_PruebaEstereografica.m**:

```
Funcion:    geo_PruebaEstereografica.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba la correcta programacion de la proyeccion estereografica

Entradas:   Ninguna

Salidas:    - Un fichero en el directorio de trabajo

Ejemplo:    geo_PruebaEstereografica;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_Elipsoide.m
                             - geo_GmsRad.m
```

```
                                - geo_PonLonGeoDominio.m
                                - geo_GdecRad.m
                                - geo_GeoEstereografica.m
                                - geo_EstereograficaGeo.m
                                - geo_RadGms.m
                                - geo_ImprimeTiempo.m

  Funciones relacionadas de GEO: - geo_PruebaTranMerc.m

  Ultima revision: 25-05-06

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaGenIntDir.m**:

- **geo_PruebaGenIntDirH.m**:

- **geo_PruebaGenIntInvH.m**:

- **geo_PruebaGeneraMalla.m**:

- **geo_PruebaGeoTri.m**:

- **geo_PruebaIntDir.m**:

- **geo_PruebaIntDirH.m**:

- **geo_PruebaIntInv.m**:

- **geo_PruebaIntInvH.m**:

- **geo_PruebaInterpolaTin.m**:

```
  Funcion:    geo_PruebaInterpolaTin.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Comprueba la correcta programacion de las funciones de
              interpolacion en triangulaciones de Delaunay
```

```
Entradas:   Ninguna

Salidas:    - Un fichero en el directorio de trabajo con el resumen de los
              resultados
            - Varios ficheros DXF en el directorio de trabajo
            - Varios ficheros en el directorio de trabajo que contienen un
              listado de los puntos utilizados en formato 'Codigo X Y Z'

Ejemplo:    geo_PruebaInterpolaTin;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_Elipsoide.m
                             - geo_Mde.m
                             - geo_Geoide.m
                             - geo_CreaMallaPuntosElipsoide.m
                             - geo_GmsRad.m
                             - geo_AltitudMdeMalla.m
                             - geo_OndGeoideMalla.m
                             - geo_GeoTranMerc.m
                             - geo_Delaunay.m
                             - geo_CurvadoTin.m
                             - geo_DibujaTin.m
                             - geo_DibujaCurvado.m
                             - geo_Dibujo.m
                             - geo_Lista.m
                             - geo_CreaCapaDibujo.m
                             - geo_DameElementoLista.m
                             - geo_CreaPuntoDibujo.m
                             - geo_CreaPolilineaDibujo.m
                             - geo_LimitesVisualDibujo.m
                             - geo_CreaDibujoDxf.m
                             - geo_ImprimeTiempo.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 13-11-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaLam.m**:

- **geo_PruebaLatitudes.m**:

```
Funcion:    geo_PruebaLatitudes.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba la correcta programacion de las distintas latitudes
            utilizadas en geodesia y cartografia matematica

Entradas:   Ninguna

Salidas:    - Un fichero en el directorio de trabajo
```

```
Ejemplo:    geo_PruebaLatitudes;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_Elipsoide.m
                             - geo_GdecRad.m
                             - geo_GmsRad.m
                             - geo_LatGeoLatConf.m
                             - geo_LatConfLatGeo.m
                             - geo_LatGeoLatAut.m
                             - geo_LatAutLatGeo.m
                             - geo_LatGeoLatRect.m
                             - geo_LatRectLatGeo.m
                             - geo_LatGeoLatRed.m
                             - geo_LatRedLatGeo.m
                             - geo_LatGeoLatGeoc.m
                             - geo_LatGeocLatGeo.m
                             - geo_LatGeoLatIso.m
                             - geo_LatIsoLatGeo.m
                             - geo_RadGms.m
                             - geo_ImprimeTiempo.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 25-05-06

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaMinCuadEcObsRegular.m**:

- **geo_PruebaPerimSup.m**:

```
Funcion:    geo_PruebaPerimSup.m

Proposito:  Comprueba la correcta programacion de las funciones de calculo del
            perimetro y la superficie de un poligono en el plano

Entradas:   - Indicador de calculo de errores. Tres posibilidades:
              - 1: No se hace calculo de errores
              - 2: Se utilizaran matrices diagonales
              - 3: Se utilizaran matrices completas
            - Indicador de activacion de control de argumentos de entrada de
              las funciones utilizadas. Dos posibilidades:
              - 0: No se controlan los argumentos de entrada
              - 1: Si se controlan los argumentos de entrada
              Por omision, se utilizara la configuracion de control de errores
              actual de la variable global

Salidas:    - Un fichero, de nombre geo_PruebaPerimSup.sal, en el directorio
              de trabajo, con los resultados de los calculos

Ejemplo:    geo_PruebaPerimSup;

Usa geoVarGlobal: Si
```

```
Original:   15-11-2006, Jose Luis Garcia Pallero, jgpallero@gmail.com

Revisiones: 22-02-2009, Jose Luis Garcia Pallero, jgpallero@gmail.com
              Puesta en forma de funcion del script de prueba original y
              documentacion
```

- **geo_PruebaProbPrinGeo.m**:

```
Funcion:    geo_PruebaProbPrinGeo.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba la correcta programacion de los problemas principales de
            la geodesia: problemas directo en inverso

Entradas:   Ninguna

Salidas:    - Un fichero en el directorio de trabajo

Ejemplo:    geo_PruebaProbPrinGeo;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_Elipsoide.m
                             - geo_QuitaEspacios.m
                             - geo_GmsRad.m
                             - geo_Fecha.m
                             - geo_RadGms.m
                             - geo_ProbDirGeo.m
                             - geo_ProbInvGeo.m
                             - geo_ImprimeTiempo.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 27-01-07

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaRadiacionPlano.m**:

```
Funcion:    geo_PruebaRadiacionPlano.m

Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

Proposito:  Comprueba la correcta programacion de las funciones para el
            calculo de una radiacion topografica

Entradas:   Ninguna

Salidas:    - Un fichero en el directorio de trabajo

Ejemplo:    geo_PruebaRadiacionPlano;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_AcimutTopografico.m
```

```
                                - geo_DistanciaPlano.m
                                - geo_DistanciaEspacio.m
                                - geo_DistanciaCenital.m
                                - geo_RadiacionPlano.m
                                - geo_RadiacionPlanoH.m
                                - geo_RadCent.m
                                - geo_ImprimeTiempo.m

  Funciones relacionadas de GEO: Ninguna

  Ultima revision: 18-09-06

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaTranMerc.m**:

```
  Funcion:    geo_PruebaTranMerc.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Comprueba la correcta programacion de la proyeccion transversa de
              Mercator (Gauss-Kruger)

  Entradas:   Ninguna

  Salidas:    - Un fichero en el directorio de trabajo

  Ejemplo:    geo_PruebaTranMerc;

  Funciones utilizadas de GEO: - geo_VarGlobal.m
                               - geo_CambiaEstadoControles.m
                               - geo_Elipsoide.m
                               - geo_GmsRad.m
                               - geo_GdecRad.m
                               - geo_GeoTranMerc.m
                               - geo_TranMercGeo.m
                               - geo_RadGms.m
                               - geo_ImprimeTiempo.m

  Funciones relacionadas de GEO: - geo_PruebaEstereografica.m

  Ultima revision: 25-05-06

  Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

- **geo_PruebaTri.m**:

- **geo_PruebaTriEnu.m**:

```
  Funcion:    geo_PruebaTriEnu.m

  Programada por: Jose Luis Garcia Pallero, jgpallero@gmail.com

  Proposito:  Comprueba la correcta programacion de las funciones de
              transformacion entre incrementos de coordenadas cartesianas
```

```
            tridimensionales geocentricas y coordenadas cartesianas
            topocentricas

Entradas:   Ninguna

Salidas:    - Un fichero en el directorio de trabajo

Ejemplo:    geo_PruebaTriEnu;

Funciones utilizadas de GEO: - geo_VarGlobal.m
                             - geo_CambiaEstadoControles.m
                             - geo_Elipsoide.m
                             - geo_QuitaEspacios.m
                             - geo_GmsRad.m
                             - geo_Fecha.m
                             - geo_RadGms.m
                             - geo_ProbDirGeo.m
                             - geo_ProbInvGeo.m
                             - geo_ImprimeTiempo.m

Funciones relacionadas de GEO: Ninguna

Ultima revision: 07-06-08

Revisada por: Jose Luis Garcia Pallero, jgpallero@gmail.com
```

# Capítulo 23
# Sección 15-documentacion

En esta carpeta se almacenan archivos relacionados con la documentación de GEO.

## 23.1. Carpeta 01-ayudaPdf

En esta carpeta se almacena la ayuda completa de GEO en formato PDF.

## 23.2. Carpeta 02-ayudaHtml

En esta carpeta se almacena la ayuda completa de GEO en formato HTML.

## 23.3. Carpeta 03-definicionSintaxis

En esta carpeta se almacenan los archivos de estilo de sintaxis para utilizar en los editores *Kate/Kwrite* en *GNU/Linux* y *CodePad* en *MS Windows*.

## 23.4. Carpeta 04-fuentesAyuda

En esta carpeta se almacenan los archivos fuente de la ayuda de GEO (a excepción de la ayuda de las funciones, que se genera de forma automática).

### 23.4.1. Carpeta 01-ficheroGlobal

En esta carpeta se almacenan los ficheros globales de los distintos tipos de ayuda de Geo. Estos ficheros serán los llamados directamente por el programa `latex` durante el proceso de compilación.

### 23.4.2. Carpeta 02-preambulo

En esta carpeta se almacenan los archivos fuente con los comandos del preámbulo del documento LaTeX de los manuales de GEO.

### 23.4.3. Carpeta 03-frontmatter

En esta carpeta se almacenan los ficheros fuente con los textos de la primera parte de los manuales de ayuda de GEO: introducción, histórico de cambios, etc.[1]

[1] El archivo fuente correspondiente a la portada de cada documento de ayuda será generado por la función `geo_PortadaAyudaLaTeX.m` de GEO.

### 23.4.4. Carpeta 04-mainmatter

En esta carpeta se almacenan los ficheros fuentes que contienen el texto de la parte principal de los distintos manuales de GEO.

#### 23.4.4.1. Carpeta 01-ficheroGlobalMainmatter

En esta carpeta se almacenan los ficheros fuente que engloban toda la sección `mainmatter` de la ayuda de GEO.

#### 23.4.4.2. Carpeta 02-fuentesUsuario

En esta carpeta se almacenan los archivos fuente de la ayuda del usuario de GEO.

#### 23.4.4.3. Carpeta 03-fuentesDesarrollador

En esta carpeta se almacenan los archivos fuente de la ayuda del desarrollador de GEO.

#### 23.4.4.4. Carpeta 04-fuentesIntroduccionCarpetas

#### 23.4.4.5. Carpeta 05-fuentesApendices

En esta carpeta se almacenan los archivos fuente de algunos capítulos de los apéndices del manual de GEO.

#### 23.4.4.6. Carpeta 06-fuentesLicencias

En esta carpeta se almacenan los archivos fuente en formato LaTeX de las licencias de GEO.

### 23.4.5. Carpeta 05-backmatter

En esta carpeta se almacenan los ficheros fuente que contienen el código necesario para generar la última parte de la ayuda de GEO: bibliografía e índices.

### 23.4.6. Carpeta 06-auxiliares

Esta carpeta contiene todos los archivos auxiliares necesarios para generar los manuales de ayuda de GEO.

#### 23.4.6.1. Carpeta 01-bibliografia

En esta carpeta se almacenan las bases de datos de BibTeX que componen la bibliografía de este documento.

#### 23.4.6.2. Carpeta 02-figuras

En esta carpeta se almacenan las figuras[2] incluidas en la ayuda de GEO.

#### 23.4.6.3. Carpeta 03-estilos

En esta carpeta se almacenan los archivos de estilo necesarios para la correcta compilación de los manuales de ayuda de GEO.

[2] En formato *.eps y *.jpg

#### 23.4.6.4. Carpeta 04-makefiles

En esta carpeta se guardan los ficheros de instrucciones de compilación de la documentación de GEO que serán leídos por el programa `make`.

## 23.5. Carpeta 05-licencias

En esta carpeta se almacenan las licencias, en formato ASCII, de GEO.

# Parte IV

# Apéndices

# Apéndice A
# Instalación de *GNU Octave*

En lo que sigue se describirá la instalación de *GNU Octave* sobre distintas plataformas.

# Apéndice B
# Editores de texto

Dado que el trabajo diario con GEO, ya sea como usuario o como desarrollador, consiste principalmente en la edición de archivos de texto, es necesario dedicar algo de tiempo a describir algunos programas que podemos utilizar para tal cometido. Principalmente nos centraremos en aquellos editores que están disponibles en un mayor número de sistemas operativos distintos, aunque también tendremos en cuenta otros que son exclusivos de algún sistema concreto. Por otra parte, también trataremos de editores con entorno gráfico o de línea de comandos exclusivamente.

## B.1. El editor *Kate*

Aquí toca hablar del editor *Kate*.

### B.1.1. Instalación de archivos de resaltado de sintaxis

Aquí contaré cómo se instalan los archivos personalizados de resaltado de sintaxis.

## B.2. El editor *vi/vim*

Aquí toca hablar del editor *vi/vim*.

### B.2.1. Instalación de archivos de resaltado de sintaxis

Aquí contaré cómo se instalan los archivos personalizados de resaltado de sintaxis.

## B.3. El editor *CodePad*

Aquí toca hablar del editor *CodePad*.

## B.4. El editor *nano*

Aquí toca hablar del editor *nano*.

## B.5. El editor *Gedit*

Aquí toca hablar del editor *Gedit*.

## B.6. El editor *jEdit*

Aquí toca hablar del editor *jEdit*.

## B.7. El editor *emacs*

Aquí toca hablar del editor *emacs*.

## B.8. El editor *Smultron*

Aquí toca hablar del editor *Smultron*.

## B.9. El editor *Geany*

Aquí toca hablar del editor *Geany*.

## B.10. El editor *SciTE*

Aquí toca hablar del editor *SciTE*.

# Apéndice C
# GNU General Public License

Version 3, 29 June 2007
Copyright © 2007 Free Software Foundation, Inc. http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we

wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

## Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries,

or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

*a*) The work must carry prominent notices stating that you modified it, and giving a relevant date.

*b*) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

*c*) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

*d*) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

*a*) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

*b*) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

*c*) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

*d*) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

*e*) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

*a*) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

*b*) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

*c*) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

*d*) Limiting the use for publicity purposes of names of licensors or authors of the material; or

*e*) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

*f*) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

    If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING,

BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

## End of Terms and Conditions

### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.

# Apéndice D
# GNU Free Documentation License

Version 1.3, 3 November 2008
Copyright© 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique

number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

## 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

## ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

# Bibliografía


[1] Andreu, M. A. y Simó, C. *Determinació del geoide UB91 a Catalunya.* N.º 1 en Monografies tècniques. Institut Cartogràfic de Catalunya, Barcelona, 1.ª ed., abril 1992. ISBN 84-393-2190-2.
URL http://www.icc.es

[2] Aranda, Tomás y García, J. Gabriel. *Notas sobre Matlab.* Servicio de Publicaciones. Universidad de Oviedo, Oviedo, 1.ª ed., 1999. ISBN 84-8317-180-5.
URL http://www.uniovi.es/publicaciones/

[3] Bravo Yuste, Santos. *Métodos matemáticos avanzados para científicos e ingenieros.* N.º 48 en Manuales UEx. Universidad de Extremadura. Servicio de Publicaciones, Cáceres, 1.ª ed., 2006. ISBN 84-689-9786-2.
URL http://www.unex.es/eweb/fisteor/santos/mma

[4] Bugayevskiy, Lev M. y Snyder, John Parr. *Map projections. A reference manual.* Taylor & Francis, Londres, 1.ª ed., 1995. ISBN 0-7484-0303-5. 3"a reimpresión, 2002.

[5] Chueca Pazos, Manuel; Herráez Boquera, José y Berné Valero, José Luis. *Tratado de topografía*, vol. 1: *Teoría de errores e instrumentación.* Editorial Paraninfo, S.A., Madrid, 1.ª ed., 1996. ISBN 84-283-2308-9.

[6] ———. *Tratado de topografía*, vol. 2: *Métodos topográficos.* Editorial Paraninfo, S.A., Madrid, 1.ª ed., 1996. ISBN 84-283-2309-7.

[7] ———. *Tratado de topografía*, vol. 3: *Redes topográficas y locales. Microgeodesia.* Editorial Paraninfo, S.A., Madrid, 1.ª ed., 1996. ISBN 84-283-2310-0.

[8] Cid Palacios, Rafael y Ferrer Martínez, Sebastián. *Geodesia geométrica, física y por satélites.* Instituto Geográfico Nacional, Madrid, 1.ª ed., 1997. ISBN 84-7819-085-6.
URL http://www.ign.es

[9] Eaton, John W. *GNU Octave. A high-level interactive language for numerical computations*, 3.ª ed., 1997.
URL www.octave.org

[10] Heiskanen, Weikko A. y Moritz, Helmut. *Geodesia física.* Instituto Geográfico Nacional, Madrid, 1.ª ed., junio 1985. ISBN 84-505-1386-3. Edición original de 1967.
URL http://www.ign.es

[11] Leick, Alfred. *GPS satellite surveying.* John Wiley & Sons, Inc, Hoboken, New Jersey, 3.ª ed., 2004. ISBN 0-471-05930-7.
URL www.wiley.com

[12] Lerma García, José Luis. *Fotogrametría moderna: analítica y digital.* Universidad Politécnica de Valencia, Valencia, 1.ª ed., 2002. ISBN 84-9705-210-2. Ref.: 2002-430.

[13] Martín Asín, Fernando. *Astronomía*. Editorial Paraninfo, S.A., Madrid, 3.ª ed., 1990. ISBN 84-300-7163-6.
URL fmasin@telefonica.net

[14] ———. *Geodesia y cartografía matemática*. Editorial Paraninfo, S.A., Madrid, 3.ª ed., 1990. ISBN 84-398-0248-X.
URL fmasin@telefonica.net

[15] ———. *Problemas de astronomía*. Editorial Paraninfo, S.A., Madrid, 4.ª ed., 1990. ISBN 84-283-1041-6.
URL fmasin@telefonica.net

[16] Martín Furones, Ángel y Padín Devesa, Jordi. *Geodesia física: material de prácticas*. Universidad Politécnica de Valencia, Valencia, 1.ª ed., 2003. ISBN 84-9705-465-2.
URL http://www.upv.es/informa/info/PUBL/index_menu200c.html

[17] Martín Furones, Ángel y Padín Devesa, Jorge. *Establecimiento de la red gravimétrica de la provincia de Valencia*. Universidad Politécnica de Valencia, Valencia, 1.ª ed., 2004. ISBN 84-9705-651-5.
URL http://www.upv.es/informa/info/PUBL/index_menu200c.html

[18] Borrell i Nogueras, Guillem. *Introducción informal a MatLab y Octave*, 2005. Versión 0.10.
URL http://torroja.dmt.upm.es:9673/Guillem_Site/

[19] Pérez López, César. *Matlab y sus aplicaciones en las ciencias y la ingeniería*. Pearson Educación, S.A., Madrid, 2002. ISBN 84-205-3537-0.
URL http://www.pearsoneducacion.com

[20] Schenk, Toni. *Fotogrametría digital*, vol. 1: *Antecedentes, fundamentos, procedimientos automáticos de orientación* de *Geomática*. Marcombo, S.A.; Instituto Cartográfico de Cataluña (ICC), Barcelona, 1.ª ed., 2002. ISBN 84-276-1330-0 (Marcombo) y 84-393-5746-X (ICC).
URL http://www.marcombo.es

[21] Snyder, John Parr. *Map projections-A working manual*. N.º 1395 en Geological Survey professional paper. U.S. Geological Survey, Washington, DC, 1.ª ed., 2003. 5"a reimpresión.

[22] Strang, Gilbert y Borre, Kai. *Linear algebra, geodesy, and GPS*. Wellesley-Cambridge Press, Wellesley, 1997. ISBN 0-9614088-6-3.
URL http://www-math.mit.edu/~{}gs

[23] Torge, Wolfgang. *Geodesy*. Walter de Gruyter, Nueva York, 3.ª ed., 2001. ISBN 3-11-017072-8.
URL http://www.degruyter.com

[24] Udías, Agustín y Mezcua, Julio. *Fundamentos de geofísica*. N.º 167 en Alianza Universidad Textos. Alianza Editorial, S.A., Madrid, 2.ª ed., 1997. ISBN 84-206-8167-9.

[25] Wolf, Paul R. y Ghilani, Charles D. *Adjustment computations: statistics and least squares in surveying and GIS*. Wiley series in surveying and boundary control. Wiley-Interscience, Nueva York, 3.ª ed., 1997. ISBN 0-471-16833-5.

[26] Zakatov, P.S. *Curso de geodesia superior*. Rubiños-1860, S.A., Madrid, 1.ª ed., 1997. ISBN 84-8041-097-3.

# Índice terminológico


**B**

bibtex . . . . . 28

**D**

dvipdfm . . . . . 28

**E**

eval . . . . . 37

**F**

feval . . . . . 37

**G**

geo_GeneraAyudaFuncionesLaTeX.m . . . . . 27
geo_GeneraAyudaLaTeX.m . . . . . 27
geo_GeneraAyudaUsuarioLaTeX.m . . . . . 27
geo_HayQueCalcularErrores.m . . . . . 35

**L**

latex . . . . . 28

**M**

makeindex . . . . . 28

**P**

pdflatex . . . . . 28

# Índice de funciones

En este índice se listan todas las funciones de GEO enlazadas a la página del documento donde se encuentra la ayuda correspondiente. El resto de lugares en los que pueda aparecer cualquier función se indican en el índice terminológico.

## G


geo_AcimutAnguloMate.m ..... 168
geo_AcimutTopografico.m ..... 223
geo_ActivaControles.m ..... 168
geo_AjustaFilasVectorMatriz.m ..... 169
geo_AjustaTexto.m ..... 170
geo_AltitudMdeMalla.m ..... 205
geo_AltitudMdeTin.m ..... 206
geo_AnalisisRedGps.m ..... 268
geo_AnguloMateAcimut.m ..... 170
geo_AsciiBin.m ..... 80
geo_Bibliografia.m ..... 166
geo_BinAscii.m ..... 81
geo_BuscaNumero.m ..... 171
geo_CalcLectDialRectaLRG.m ..... 296
geo_CalculoRedGps.m ..... 270
geo_CambiaEstadoControles.m ..... 194
geo_CentGdec.m ..... 52
geo_CentGms.m ..... 52
geo_CentRad.m ..... 53
geo_ChiCuadInv.m ..... 71
geo_CoeficienteCorrelacion.m ..... 71
geo_ComentariosFichero.m ..... 171
geo_CompilaOct.m ..... 172
geo_CompruebaColumnas.m ..... 173
geo_Concatena.m ..... 173
geo_ControlDominio.m ..... 48
geo_ControlDominioAF.m ..... 48
geo_ControlDominioDiaMes.m ..... 49
geo_ControlDominioInterpMalla.m ..... 49
geo_ControlDominioLatGeoAF.m ..... 50
geo_ControlDominioLatGeoLonGeo.m ..... 50
geo_ControlDominioLatGeoLonGeoHElip.m ..... 51
geo_CopiaArchivo.m ..... 174
geo_Copyright.m ..... 168
geo_CreaBloqueDibujo.m ..... 102
geo_CreaCapaDibujo.m ..... 103
geo_CreaCapaDxf.m ..... 131
geo_CreaCircunferenciaDibujo.m ..... 104
geo_CreaDefinicionBloqueDibujo.m ..... 105
geo_CreaDefinicionBloqueDxf.m ..... 132
geo_CreaDibujoDxf.m ..... 134
geo_CreaElipseDibujo.m ..... 106
geo_CreaHeaderDxf.m ..... 135
geo_CreaLineaDibujo.m ..... 107
geo_CreaLineaPuntosElipsoide.m ..... 174
geo_CreaLineaPuntosPlano.m ..... 175
geo_CreaMallaPuntosElipsoide.m ..... 176
geo_CreaMallaPuntosPlano.m ..... 177
geo_CreaMatricesCompRedGps.m ..... 274
geo_CreaPolilineaDibujo.m ..... 108
geo_CreaPuntoDibujo.m ..... 109
geo_CreaSimboloDibujo.m ..... 110
geo_CreaTextoDibujo.m ..... 111
geo_CreaTipoLineaDxf.m ..... 135
geo_CurvadoTin.m ..... 94
geo_DameElementoLista.m ..... 57
geo_Delaunay.m ..... 95
geo_DesactivaControles.m ..... 178
geo_DibujaBloqueDxf.m ..... 136
geo_DibujaCircunferenciaDxf.m ..... 137
geo_DibujaElipseDxf.m ..... 138
geo_DibujaLineaDxf.m ..... 139
geo_DibujaPolilineaDxf.m ..... 140
geo_DibujaPuntoDxf.m ..... 141
geo_DibujaSimboloDxf.m ..... 142
geo_DibujaTextoDxf.m ..... 143
geo_Dibujo.m ..... 112
geo_DistanciaCenital.m ..... 242
geo_DistanciaEspacio.m ..... 242

geo_DistanciaPlano.m ..... 224
geo_DistMerLatGeo.m ..... 247
geo_DominioAngular.m ..... 179
geo_EfemeridesSimon.m ..... 300
geo_ElipseErrorPlano.m ..... 76
geo_Elipsoide.m ..... 197
geo_EnuTri.m ..... 210
geo_EsBisiesto.m ..... 299
geo_EsColorGrosor.m ..... 115
geo_EsColorTipoGrosor.m ..... 115
geo_EsColorTipoVis.m ..... 116
geo_EsDibujo.m ..... 116
geo_EsElipsoide.m ..... 197
geo_EsEntero.m ..... 43
geo_EsEscalar.m ..... 43
geo_EsEstructuraMalla.m ..... 82
geo_EsFichero.m ..... 44
geo_EsGeoide.m ..... 199
geo_EsLista.m ..... 57
geo_EsListaCapas.m ..... 117
geo_EsMatriz.m ..... 44
geo_EsMatrizSimetrica.m ..... 45
geo_EsMde.m ..... 203
geo_EsNatural.m ..... 46
geo_EstadoControles.m ..... 195
geo_EstereograficaGeo.m ..... 259
geo_EsTexto.m ..... 46
geo_EsTipoAlmGeo.m ..... 77
geo_EsTipoBinarioGeo.m ..... 77
geo_EstructuraMalla.m ..... 82
geo_EsVectorColumna.m ..... 47
geo_ExisteEnLista.m ..... 58
geo_ExisteTextoEnFichero.m ..... 179
geo_ExtraeAyudaFichero.m ..... 180
geo_ExtraeAyudaLaTeX.m ..... 162
geo_ExtraeDatosFicheroMalla.m ..... 84
geo_ExtraeVentanaMalla.m ..... 84
geo_Fecha.m ..... 180
geo_FechaDiaJuliano.m ..... 299
geo_FicheroConocido.m ..... 181
geo_Find.m ..... 181
geo_FInv.m ..... 72
geo_FuncAsocLegendre.m ..... 281
geo_FuncionesCodePad.m ..... 182
geo_FuncionesGeo.m ..... 183
geo_FuncionesGeoResaltado.m ..... 183
geo_FuncionesKate.m ..... 184
geo_FuncionesNano.m ..... 184
geo_FuncionesVim.m ..... 185
geo_GdecCent.m ..... 53
geo_GdecGms.m ..... 53
geo_GdecRad.m ..... 54
geo_GeneraAyudaFuncionesLaTeX.m ..... 163
geo_GeneraAyudaLaTeX.m ..... 164
geo_GeneraAyudaUsuarioLaTeX.m ..... 164
geo_GeneraPorHacerLaTeX.m ..... 165
geo_GenRedGPS.m ..... 267
geo_GenRedPlanoCla.m ..... 268
geo_GeoEstereografica.m ..... 261
geo_Geoide.m ..... 199
geo_GeoTranMerc.m ..... 262
geo_GeoTri.m ..... 211
geo_GmsCent.m ..... 54
geo_GmsGdec.m ..... 55
geo_GmsRad.m ..... 55
geo_GravedadCilindroVertical.m ..... 283
geo_GravedadEsfera.m ..... 284
geo_GravedadPrismaMacMillan.m ..... 286
geo_GravedadPrismaNagy.m ..... 288
geo_GravedadPunto.m ..... 289
geo_GravedadRectangulo.m ..... 291
geo_GravedadSomigliana.m ..... 293
geo_HayQueCalcularErrores.m ..... 185
geo_HayRepeticionesEnLista.m ..... 58
geo_IgualesDistintoDeValor.m ..... 186
geo_ImprimeTiempo.m ..... 186
geo_IndGeoVarGlobal.m ..... 187
geo_IntDirectaPlano.m ..... 225
geo_IntDirectaPlanoH.m ..... 227
geo_IntDosDistanciasPlano.m ..... 229
geo_InterpolaInvDistEspacio.m ..... 87
geo_InterpolaInvDistPlano.m ..... 88
geo_InterpolaMalla.m ..... 86
geo_InterpolaTin.m ..... 90
geo_InterpolaTriangulo.m ..... 91
geo_IntInversaPlano.m ..... 231
geo_IntInversaPlanoH.m ..... 233
geo_IntTresDistanciasPlano.m ..... 235
geo_InvierteLista.m ..... 59
geo_LatAutLatGeo.m ..... 215
geo_LatConfLatGeo.m ..... 216
geo_LatGeocLatGeo.m ..... 221
geo_LatGeoDistMer.m ..... 248
geo_LatGeoLatAut.m ..... 216
geo_LatGeoLatConf.m ..... 217
geo_LatGeoLatGeoc.m ..... 218
geo_LatGeoLatIso.m ..... 218
geo_LatGeoLatRect.m ..... 219
geo_LatGeoLatRed.m ..... 220
geo_LatIsoLatGeo.m ..... 221
geo_LatRectLatGeo.m ..... 222
geo_LatRedLatGeo.m ..... 223
geo_LeeDirectorio.m ..... 188
geo_LeeLineaFichero.m ..... 188

geo_LimitesVisualDibujo.m ..... 118
geo_Lista.m ..... 59
geo_LongitudLista.m ..... 60
geo_LonTipoBinarioGeo.m ..... 78
geo_MatrizDiagMatrizComp.m ..... 66
geo_MatrizSimVector.m ..... 67
geo_MatrizSimVectorInterna.m ..... 68
geo_MatrizVector.m ..... 61
geo_Mde.m ..... 203
geo_MediaCoordenadas.m ..... 97
geo_MinCuadConstrEcObs.m ..... 98
geo_MinCuadEcCond.m ..... 99
geo_MinCuadEcObs.m ..... 99
geo_MinCuadModMixto.m ..... 100
geo_NodosCircunferencia.m ..... 119
geo_NodosCuadrado.m ..... 120
geo_NodosCurvado.m ..... 122
geo_NodosElipse.m ..... 123
geo_NodosEstrella.m ..... 125
geo_NodosLinea.m ..... 126
geo_NodosTin.m ..... 128
geo_NodosTriangEq.m ..... 129
geo_NormInv.m ..... 73
geo_NumeroVisualesRedGps.m ..... 272
geo_ObservacionesLibresRed.m ..... 276
geo_OndGeoideMalla.m ..... 201
geo_OrdenaMatriz.m ..... 62
geo_OrdenaMatrizPlantilla.m ..... 63
geo_ParteZonaEnPrismas.m ..... 295
geo_Path.m ..... 189
geo_PerimetroPlano.m ..... 236
geo_PolLegendre.m ..... 282
geo_PonAcimutDominio.m ..... 209
geo_PonLatGeoDominio.m ..... 209
geo_PonLonGeoDominio.m ..... 210
geo_PortadaAyudaLaTeX.m ..... 166
geo_PosFicheroLineaDatos.m ..... 190
geo_PosGruposVector.m ..... 64
geo_PrismaPuntoMasa.m ..... 294
geo_ProbDirGeo.m ..... 248
geo_ProbDirGeoElip.m ..... 250
geo_ProbDirGeoEsf.m ..... 244
geo_ProbInvGeo.m ..... 252
geo_ProbInvGeoElip.m ..... 253
geo_ProbInvGeoEsf.m ..... 245
geo_PuntosLibresRed.m ..... 273
geo_QuitaEspacios.m ..... 190
geo_QuitaLeyendaAtribDibujo.m ..... 131
geo_RadCent.m ..... 56
geo_RadGdec.m ..... 56
geo_RadGms.m ..... 56
geo_RadiacionPlano.m ..... 237
geo_RadiacionPlanoH.m ..... 239
geo_RadioElipMer.m ..... 255
geo_RadioParalelo.m ..... 256
geo_RadioPrimVert.m ..... 256
geo_Redimensiona.m ..... 65
geo_RepiteEnLista.m ..... 60
geo_RestauraControles.m ..... 191
geo_ResTriInf.m ..... 100
geo_ResTriSup.m ..... 101
geo_Subcadena.m ..... 192
geo_SuperficiePlano.m ..... 240
geo_TauInv.m ..... 74
geo_TestFuncionesGeo.m ..... 192
geo_TestGlobModChiCuad.m ..... 74
geo_TInv.m ..... 74
geo_TipoAlmGeoTipoAlmOctave.m ..... 78
geo_TipoBinGeoTipoBinC.m ..... 79
geo_TipoBinGeoTipoBinOctave.m ..... 79
geo_TipoDato.m ..... 47
geo_TranMercGeo.m ..... 264
geo_TriEnu.m ..... 212
geo_TriGeo.m ..... 214
geo_UdsTrabajoUdsBase.m ..... 195
geo_UdsTrabajoUdsBaseVar.m ..... 196
geo_UneAlista.m ..... 61
geo_UneMatricesDiagonal.m ..... 68
geo_UneMatrizVectorDiagonal.m ..... 69
geo_VarGeoVarGlobal.m ..... 193
geo_VarGlobal.m ..... 41
geo_VectorMatriz.m ..... 66
geo_VectorMatrizSim.m ..... 69
geo_VectorMatrizSimInterna.m ..... 70
geo_WarningOct.m ..... 193

# Índice de funciones compiladas

En este índice se listan todas las funciones de GEO escritas en *C++* o *Fortran* enlazadas a la página del documento donde se encuentra la ayuda correspondiente. El resto de lugares en los que pueda aparecer cualquier función se indican en el índice terminológico.

**G**


geo_AsciiBinOct.cc ..... 149
geo_ExtraeDatosFicheroMallaOct.cc ..... 150
geo_FuncAsocLegendreSoloOct.cc ..... 151
geo_FuncAsocLegendreTodoOct.cc ..... 152
geo_GravedadPrismaOct.cc ..... 152
geo_GravedadRectanguloOct.cc ..... 153
geo_InterpolaMallaOct.cc ..... 154
geo_PolLegendreOct.cc ..... 155
geo_ProbDirGeoElipOct.cc ..... 156
geo_ProbDirGeoEsfOct.cc ..... 156
geo_ProbInvGeoElipOct.cc ..... 157
geo_ProbInvGeoEsfOct.cc ..... 158
geo_ResTriInfOct.cc ..... 158
geo_ResTriSupOct.cc ..... 159
geo_TestEndianOct.cc ..... 160
geo_TriangleOct.cc ..... 160

# Índice de ficheros C/C++, Fortran y Bash

En este índice se listan todos los ficheros de GEO escritos en *C/C++*, *Fortran* o *Bash* enlazados a la página del documento donde se encuentra la ayuda correspondiente. El resto de lugares en los que pueda aparecer cualquier fichero se indican en el índice terminológico.


**C**

`compilaOct.sh` ..... 162
`constantesOct.h` ..... 145

**D**

`datosBinariosOct.cc` ..... 147
`datosBinariosOct.h` ..... 145
`defineTriangle.h` ..... 145

**F**

`funcionesAuxiliaresOct.cc` ..... 147
`funcionesAuxiliaresOct.h` ..... 145

**G**

`geodesiaOct.cc` ..... 148
`geodesiaOct.h` ..... 145
`gravimetriaOct.cc` ..... 148
`gravimetriaOct.h` ..... 146

**M**

`matematicasOct.cc` ..... 148
`matematicasOct.h` ..... 146

**R**

`rutaLibOct.h` ..... 146

**T**

`triangle.c` ..... 148
`triangle.h` ..... 146
`triangleOct.cc` ..... 149
`triangleOct.h` ..... 147

# Índice de ficheros de datos

En este índice se listan todos los ficheros de configuración de GEO enlazados a la página del documento donde se encuentra la ayuda correspondiente. El resto de lugares en los que pueda aparecer cualquier fichero se indican en el índice terminológico.


**B**

`bibliografia.dat` ..... 305

**D**

`datum.dat` ..... 306

**E**

`elipsoide.dat` ..... 306

**G**

`geoide.dat` ..... 307

**I**

`instrumento.dat` ..... 311

**M**

`mde.dat` ..... 312

**P**

`proyCarto.dat` ..... 316

**T**

`transfDatum.dat` ..... 316

# Índice de programas de prueba

En este índice se listan todos los programas de prueba de GEO enlazados a la página del documento donde se encuentra la ayuda correspondiente. El resto de lugares en los que pueda aparecer cualquier programa se indican en el índice terminológico.

**G**


geo_PruebaCalcOndulaciones.m ..... 317
geo_PruebaCalcRedGps.m ..... 317
geo_PruebaComparaGeoides.m ..... 317
geo_PruebaCurvado.m ..... 318
geo_PruebaDibujoGeomadrid.m ..... 318
geo_PruebaDosDist.m ..... 318
geo_PruebaDxf.m ..... 318
geo_PruebaEspacio.m ..... 319
geo_PruebaEspacioError.m ..... 319
geo_PruebaEspacioErrorDef.m ..... 319
geo_PruebaEstereografica.m ..... 319
geo_PruebaGeneraMalla.m ..... 320
geo_PruebaGenIntDir.m ..... 320
geo_PruebaGenIntDirH.m ..... 320
geo_PruebaGenIntInvH.m ..... 320
geo_PruebaGeoTri.m ..... 320
geo_PruebaIntDir.m ..... 320
geo_PruebaIntDirH.m ..... 320
geo_PruebaInterpolaTin.m ..... 320
geo_PruebaIntInv.m ..... 320
geo_PruebaIntInvH.m ..... 320
geo_PruebaLam.m ..... 321
geo_PruebaLatitudes.m ..... 321
geo_PruebaMinCuadEcObsRegular.m ..... 322
geo_PruebaPerimSup.m ..... 322
geo_PruebaProbPrinGeo.m ..... 323
geo_PruebaRadiacionPlano.m ..... 323
geo_PruebaTranMerc.m ..... 324
geo_PruebaTri.m ..... 324
geo_PruebaTriEnu.m ..... 324